Journal of Machine Learning Research 22 (2021) 1-41 Submitted 1/19; Revised 11/20; Published 8/21

Differentially Private Regression and Classification with Sparse Gaussian Processes

Michael Thomas Smith[1] M.T.SMITH@SHEFFIELD.AC.UK
Mauricio A. Álvarez MAURICIO.ALVAREZ@SHEFFIELD.AC.UK
Department of Computer Science
University of Sheffield, UK
Neil D. Lawrence[2] NDL21@CAM.AC.UK
Department of Computer Science and Technology
University of Cambridge, UK

Editor: Frank Wood

Abstract

A continuing challenge for machine learning is providing methods to perform computation on data while ensuring the data remains private. In this paper we build on the provable privacy guarantees of differential privacy which has been combined with Gaussian processes through the previously published *cloaking method*, an approach that tackles the problem of providing privacy for the *outputs* of a training set. In this paper we solve several shortcomings of this method, starting with the problem of predictions in regions with low data density. We experiment with the use of inducing points to provide a sparse approximation and show that these can provide robust differential privacy in outlier areas and at higher dimensions. We then look at classification, and modify the Laplace approximation approach to provide differentially private predictions. We then combine this with the sparse approximation and demonstrate the capability to perform classification in high dimensions. We finally explore the issue of hyperparameter selection and develop a method for their private selection. This paper and associated libraries provide a robust toolkit for combining differential privacy and Gaussian processes in a practical manner.

Keywords: Gaussian processes, differential privacy, classification, sparse Gaussian processes

1. Introduction

A common challenge in machine learning is balancing accuracy with privacy preservation when computing a prediction or performing inference with a dataset that contains personal data. Besides obvious examples such as household surveys and medical records, personal data is often found in other datasets, including taxi tracking (Tockar, 2014), in crowd-sourced mobile air pollution sensors (Yang et al., 2015), in road traffic incidents (Sweeney, 2015) and even in biological surveying where individuals may start transects at their own home, for example in the BeeWalk survey scheme (Comont and Miles, 2019). Simple 'anonymisation' (e.g. removing names and addresses) has been found to be insufficient for protecting the privacy of individuals when releasing statistics from a dataset (Sweeney, 1997; Ganta et al.,

1. Corresponding author.
2. Work conducted while at the University of Sheffield.

2008). This is because side information can be used to infer identity from the remaining data. Sweeney's work in particular demonstrates how combining 'anonymised' medical records with newspaper reports allows one to infer the identity of patients (Sweeney, 2015).

1.1 Differential privacy and Gaussian processes

Even aggregated statistics could reveal individual information, if, for example, one were able to query the database before and after the individual were added, the difference between the statistics could reveal private information. It is this idea of difference that leads to *differential privacy*. Assuming that the statistic is non-trivial (i.e. reports something related to the contents of the database), the only way that this difference can be masked is by adding some randomness to the result. Differential privacy (DP) is a formalism that bounds the amount that an attacker can infer about a database record from such randomised responses. Specifically, it bounds how much the probability of any given output can change upon the adding or perturbing of an individual.

Gaussian process (GP) regression is a data-efficient non-parametric method which allows the user to introduce strong priors around the data's structure. For example the smoothness of a function and the correlations between inputs. It is this correlation structure that allows one to make predictions (with uncertainty) at new, test locations: A GP assumes the variables at any subset of locations will have a Gaussian distribution (known as the prior). Prediction consists of conditioning this distribution on the observations to produce what is known as the *posterior distribution*. Due to the unique properties of the Gaussian, this distribution is also Gaussian. Its mean and covariance can be computed in closed form in a single step. We believe these features of smoothness and closed-form single-step inference are what make it possible to apply DP to GP so effectively. We will introduce Gaussian processes and differential privacy more rigorously later.

1.2 The cloaking mechanism and our privacy model

This paper builds on Smith et al. (2018) which introduced a method for differentially private Gaussian process regression, called the *cloaking mechanism*, in which they corrupted a Gaussian processes' posterior mean in order to make aspects of the training data private. **Our Privacy Model**: Specifically this method protects the *outputs* of the training data, but *requires the inputs to remain unprotected*. Although this limits the scope to a subset of data-privacy challenges, the subset is large and contains important examples. One obvious example application might be inference over census or household survey data. The existence of a property at a location (easting and northing), $\boldsymbol{x_i}$, is not private. However a feature, y_i (e.g. disease status, income, etc), about the household who live there is. Hence the training inputs, $\mathbf{X}$, to the algorithm are already public while the training outputs, $\boldsymbol{y}$, are private. More broadly, we are interested in any situation in which a population has been aggregated or organised along axes that are public. A second example described by Smith et al. (2018) involves the analysis of road collision data in Kampala, Uganda. The times and locations in $\mathbf{X}$ are public knowledge, but the ages and genders of the victims are kept private (in $\boldsymbol{y}$). A typical question that we can answer in this framework could be 'where and when are children most likely to be involved in collisions?' One can use the differentially private

inference approach in this paper and Smith et al. (2018) to compute this useful result, while ensuring that information about the victims is kept private.

This restriction allows us to take advantage of the form of the posterior of Gaussian process regression. Specifically the output vector, $\boldsymbol{y}$, only appears in the posterior *mean* and is simply linearly transformed, which meant that Smith et al. (2018) were able to propose a straightforward method for producing relatively efficient DP noise. We will introduce the details and derivation of the cloaking mechanism later.

1.3 This paper's contribution

Smith et al. (2018) identified several shortcomings or areas for future work. Specifically their method was susceptible to outliers and produced predictions with unsatisfactory levels of noise added to provide DP. Second, the method was only applicable to the Gaussian likelihood problem of regression, and not to the issue of classification. Third, although briefly discussed in the supplementary, hyperparameter optimisation was not fully handled.

In the current paper we suggest solutions to some of that method's shortcomings, specifically we propose non-stationary covariance functions to reduce the impact of outliers, and develop a method for using the Laplace approximation to perform differentially private classification. We describe a method for performing hyperparameter optimisation and finally combine a sparse approximation with the classification result to allow the algorithm to function on a high dimensional classification problem.

1.4 Paper Structure

The paper is organised as follows: In Section 2 we introduce Gaussian processes for regression, differential privacy and the cloaking method, before briefly discussing the wider context of differential privacy and non-parametric methods. In Sections 3 - 5 we introduce the three topics, nonstationary covariance, classification and parameter selection respectively. In Section 6 we describe a series of experiments: Sections 6.1 and 6.2 investigate using nonstationary covariances for reducing DP noise, while Section 6.3 looks in depth at why variable lengthscales fail to achieve the improvements expected. Sections 6.4 and 6.5 look at DP GP classification. Section 6.6 combines sparse approximation with classification. Finally, Section 6.7 demonstrates how DP parameter selection might be performed.

2. Related Work

We will first introduce Gaussian processes and differential privacy. Then we describe the cloaking mechanism from Smith et al. (2018) that provides a method for making the posterior prediction of GP regression differentially private. Finally we briefly explore the literature at the intersection of differential privacy and non-parametric methods, in an attempt to explain why GPs are well suited to use with differential privacy.

2.1 Gaussian Processes

A Gaussian process is stochastic process in which any finite subset of random variables will have a multivariate normal distribution. Figure 1 illustrates this structure. Every pair of variables, $(\boldsymbol{x}, \boldsymbol{x}')$, has a covariance defined by the kernel function, $k(\boldsymbol{x}, \boldsymbol{x}')$. A common choice

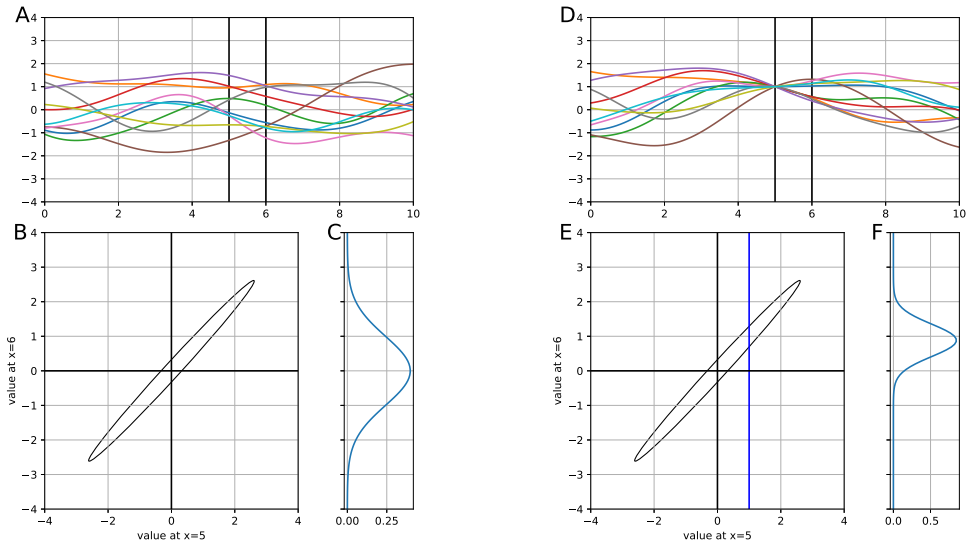

Figure 1: Gaussian process regression. (A) Samples from a GP prior (with a exponentiated quadratic kernel). (B) Covariance between a pair of locations, marked with vertical black lines in (A). (C) Distribution of $f(x = 6)$. (D) Samples from the posterior of the same GP conditioned on an observation (at $x = 5, y = 1$). (E) Illustrates how one makes a prediction at $f(x = 6|X, y)$ by conditioning on the observation. (F) Distribution of the posterior of $f(x = 6|X, Y)$. Notice how it differs from the prior in (C).

of kernel function is one in which the covariance is greater when $\boldsymbol{x}$ and $\boldsymbol{x}'$ are similar, however more complex choices can be constructed, for example to model periodic or polynomial priors. In Figure 1 we have plotted samples from the exponentiated quadratic (EQ) kernel, which in one-dimension is $k(x, x') = \sigma^2 \exp(-\frac{(x-x')^2}{2l^2})$, with parameters σ^2 and l describing the output variance and lengthscale, respectively.

To perform Gaussian process regression, one simply conditions on some of the variables (those that are observed). To be specific; we are given a set of N training input-output pairs, $\mathbf{X} \in \mathbb{R}^{N \times D}$ and $\mathbf{y} \in \mathbb{R}^N$ respectively. We assume that each output y_i is a noisy observation of a hidden, latent function f at $\boldsymbol{x}_i$, such that the added noise is independent and Gaussian.

$$y_i = f(\mathbf{x}_i) + \epsilon_i \quad \text{where } \epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

This is the likelihood model. It describes how likely we consider an output to be, given the value of $f(\boldsymbol{x}_i)$. We wish to estimate the values $\boldsymbol{f}_*$ of the latent function f at test points $\mathbf{X}_*$. We have assumed that f is drawn from a Gaussian process. The definition of a Gaussian process means that we can write the joint distribution of the observations and these test points as,

$$\begin{bmatrix} \mathbf{y} \\ \boldsymbol{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} + \sigma^2\mathbf{I} & \mathbf{K}_{*f} \\ \mathbf{K}_{f*} & \mathbf{K}_{**} \end{bmatrix}\right).$$

Where $\mathbf{K}_{**}$ is the kernel covariance between test points, $\mathbf{K}_{*f}$ is the covariance between test and training points and $\mathbf{K}_{f*}$ is its transpose. $\mathbf{K}$ is the covariance between training points. We have assumed that the data has a zero mean, achieved usually by subtracting the mean of the observations from the data, or by including a bias kernel in the prior. The conditional distribution of $f(\mathbf{X}_*)|\mathbf{y}$ can now be expressed analytically as a normal distribution with mean and covariance;

$$\bar{\mathbf{f}}_* = \mathbf{K}_{*f}(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y}$$

and

$$\text{cov}(\mathbf{f}_*) = \mathbf{K}_{**} - \mathbf{K}_{*f}(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{K}_{f*}.$$

Hyperparameter optimisation aside, one can immediately see from these definitions that in GP regression the posterior covariance (at test points) only depends on the locations of the *inputs* of the training and test data ($\mathbf{X}$ and $\mathbf{X}_*$), the private observed outputs $\boldsymbol{y}$ are not involved. The mean however does depend on $\boldsymbol{y}$ and so it is this that needs randomisation to achieve different privacy, in our setting in which $\boldsymbol{y}$ is private and $\mathbf{X}$ is public.

We refer to the expression $\mathbf{K}_{*f}(\mathbf{K} + \sigma^2\mathbf{I})^{-1}$ as the *cloaking matrix*,

$$\mathbf{C} = \mathbf{K}_{*f}(\mathbf{K} + \sigma^2\mathbf{I})^{-1}. \tag{1}$$

This defines how the posterior means, $\bar{\mathbf{f}}_*$, of the test points change with respect to changes in the private training data observations, $\boldsymbol{y}$.

2.2 Differential Privacy

We use in this paper the widely used approximate (ϵ, δ)-differential privacy, which is defined as follows. From Dwork and Roth (2014), to query a database in a DP manner, a randomised algorithm R is (ε, δ)-differentially private if, for all possible query outputs m and for all

neighbouring databases D and D' (those databases which only differ by one individual or one row),

$$P\Big(R(D) \in m\Big) \leq e^{\varepsilon} P\Big(R(D') \in m\Big) + \delta. \tag{2}$$

This simply states that we require each possible output to have almost the same probability regardless of the value of one row or individual. ε puts a bound on how much privacy is lost by releasing the result, with a smaller ε meaning more privacy. Ignoring δ, it puts a bound on the ratio (given two neighbouring databases) of the probabilities of returning any given output. If ε is very small then the two probabilities must be very similar and therefore any given output is more limited in its capacity to leak information about a row's perturbation. δ says this inequality only holds with probability $1 - \delta$. This additional term is necessary as the ratio at an arbitrary location in the densities of a pair of mean-shifted Gaussians is unbounded (as one moves further from their means the ratio will grow indefinitely) however in the regions where the ratio is large the actual probability of such an output is small. By accepting this small probability (defined by δ) we are able to use Gaussian noise by applying this approximate DP approach.

To briefly illustrate differential privacy and this additive noise, we consider a summary statistic $f(D)$ of a small hospital's patients database, D, that simply returns the number of insulin-dependent inpatients. On two days a drug supplier queries this to determine the supply of insulin. If between the queries one new patient arrives, the difference between the queries will reveal whether this individual is diabetic. We will use the Gaussian mechanism (which adds Gaussian noise) to 'corrupt' the count and protect the individual's privacy. To use any DP mechanism we first need to determine the sensitivity of the function. In this case the l_2 sensitivity, $\Delta_2(f)$, is 1, as an individual can, at most, change the total count by one. It is important to note that this sensitivity is independent of the data in the database and represents the worst case. In general,

$$\Delta_2(f) = \max_{D \sim D'} ||f(D) - f(D')||_2.$$

The Gaussian mechanism adds a sample from a Gaussian distribution with mean zero and variance $\sigma^2 \geq c\Delta_2(f)/\varepsilon$, where $c^2 > 2\log(1.25/\delta)$. So for (1,0.01)-DP in the hospital example we would need to sample from a Gaussian with a variance of at least $\sigma^2 \geq 1.77^2$. A proof for the Gaussian mechanism exists in the appendix of Dwork and Roth (2014). In particular how the inequality for c^2 is derived.

We are interested in performing regression and returning a prediction at *multiple* locations. For this we turn to the work of Hall et al. (2013) who provide a mechanism for adding noise to a vector of statistics. Rather than add a sample from a scaled unimodal Gaussian they propose that one adds a sample from a scaled *multivariate* Gaussian, $\mathbf{z} \sim \mathcal{N}_d(0, \mathbf{M})$, with mean zero and covariance $\mathbf{M}$,

$$\tilde{\mathbf{f}}_* = \mathbf{f}_* + \frac{\mathrm{c}(\delta)\Delta}{\varepsilon}\mathbf{z},$$

where $\mathbf{f}_*$ and $\tilde{\mathbf{f}}_*$ are the private and corrupted predictions, respectively. (ε, δ)-DP is achieved if $\mathrm{c}(\delta) \geq \sqrt{2\log\frac{2}{\delta}}$.

The sensitivity needs computing as in any DP mechanism, but in this case depends on the covariance of this Gaussian, as the amount it needs scaling depends on whether the covariance of the noise reflects the covariance structure in how the predictions vary with individual training points. Specifically,

$$\sup_{D\sim D'} \|\mathbf{M}^{-1/2}(\mathbf{f}_* - \mathbf{f}'_*)\|_2 \leq \Delta. \tag{3}$$

This says that the sensitivity, Δ, is a bound on the scale of the output change, in term of its Mahalanobis distance with respect to the added noise covariance.

To visualise, imagine the output vector consists of fifty copies of the same statistic. If one were to add uncorrelated noise ($\mathbf{M} = \mathbf{I}$) to the fifty values one would need to scale the noise by the l_2 norm of $\mathbf{f}_* - \mathbf{f}'_*$ which is $\sqrt{50}$ times the sensitivity of one. One can see that this stops the attacker from averaging them to find the private value. However, if one added the *same* noise sample to each (i.e. so it was highly correlated with $\mathbf{M}$ being almost an all-ones matrix), it wouldn't need scaling, as averaging wouldn't help and the covariance of the noise added would be in the same direction as the covariance of the data. Conversely if 50 independent statistics are reported the best choice of noise to add would be for it to be independent. Adding correlation between elements of the noise vector would lead to a larger value of Δ and less efficiency.

2.3 The Cloaking Method

In Gaussian process regression we can clearly see that pairs of predictions are going to be correlated, Smith et al. (2018) used the mechanism above to chose appropriate Gaussian noise to reflect this correlation.

We already defined above the cloaking matrix in (1), $\mathbf{C} = \mathbf{K}_{*f}(\mathbf{K}+\sigma^2\mathbf{I})^{-1}$, which defines how the posterior mean at the test points is computed from the values of the training points,

$$\bar{\mathbf{f}}_* = \mathbf{C}\mathbf{y}.$$

We assume one output of a training item i has been perturbed, by γ; $y'_i = y_i + \gamma$. The change in the predictions is therefore dependent on only one column of $\mathbf{C}$, $\mathbf{c}_i$; $\mathbf{f}'_* - \mathbf{f}_* = \gamma\mathbf{c}_i$ This can be substituted into the bound on Δ in (3).

$$\sup_{D\sim D'} \|\mathbf{M}^{-1/2}(\gamma\mathbf{c}_i)\|_2 \leq \Delta.$$

We expand and rearrange the norm, using the symmetry of $\mathbf{M}$. We define that γ, the change in a training value's output, is bounded by $d \geq 0$ such that $|\gamma| \leq d$. Note that the only difference between D and D' is the addition of γ.

$$\begin{aligned}
\sup_{D\sim D'} \|\mathbf{M}^{-1/2}(\gamma\mathbf{c}_i)\|_2^2 &= \sup_{D\sim D'} \left(\mathbf{M}^{-1/2}(\gamma\mathbf{c}_i)\right)^\top \left(\mathbf{M}^{-1/2}(\gamma\mathbf{c}_i)\right) \\
&= \sup_{D\sim D'} \gamma^2\mathbf{c}_i^\top\mathbf{M}^{-1}\mathbf{c}_i \\
&= d^2\mathbf{c}_i^\top\mathbf{M}^{-1}\mathbf{c}_i \\
&\leq \Delta^2.
\end{aligned}$$

We want to find $\mathbf{M}$ such that the noise sample Z is small but also that Δ is minimised. We need a way of defining what 'small' is. For this we describe the noise scale using its differential entropy, which is (ignoring an added constant) proportional to the log of the determinant: $\frac{1}{2}\ln\left((2\pi e)^k \cdot |\mathbf{\Sigma}|\right)$. Finding the value of $\mathbf{M}$ and Δ that minimises this noise scale is an underdetermined system as the scaling provided by Δ and the magnitude of the noise covariance $\mathbf{M}$ lead to equivalent outcomes. Similarly d is a constant that can be introduced later during sampling. We therefore bound $\Delta \leq 1$, and fix $d = 1$, then find $\mathbf{M}$ accordingly. In reality we then compute Δ at the end, as the gradient descent algorithm we are about to describe will not find the precise, exact solution.

We will express our problem as trying to *maximise* the entropy of a k-dimensional Gaussian distribution with covariance $\mathbf{P} = \mathbf{M}^{-1}$;

$$\text{Maximise } \ln\left(|\mathbf{P}|\right), \text{ subject to } n \text{ constraints, } 0 \leq \mathbf{c}_i^\top \mathbf{P}\mathbf{c}_i \leq 1.$$

We first consider just the upper bounds and express this as a constraint satisfaction problem using Lagrange multipliers,

$$L = \ln|\mathbf{P}| + \sum_i \lambda_i (1 - \mathbf{c}_i^\top \mathbf{P}\mathbf{c}_i). \tag{4}$$

We also have the slackness conditions (for all i), $\lambda_i(\mathbf{c}_i^\top \mathbf{P}\mathbf{c}_i - 1) = 0$ and we also require that $\lambda_i \geq 0$.

Differentiating (and setting to zero),

$$\frac{\partial L}{\partial \mathbf{P}} = \mathbf{P}^{-1} - \sum_i \lambda_i \mathbf{c}_i \mathbf{c}_i^\top = 0. \tag{5}$$

$$\mathbf{P}^{-1} = \sum_i \lambda_i \mathbf{c}_i \mathbf{c}_i^\top. \tag{6}$$

Note that as $\lambda_i \geq 0$, $\mathbf{P}^{-1}$ is positive semi-definite (psd),[1] thus the initial lower bound (that $\mathbf{c}_i^\top \mathbf{P}\mathbf{c}_i \geq 0$) is met. The upper bound (that $\mathbf{c}_i^\top \mathbf{P}\mathbf{c}_i \leq 1$) is achieved if the λ_i are correctly chosen, such that the Lagrange and slackness conditions are met.

We now must maximise the expression for L wrt λ_j. To assist with this we rewrite our expression for

$$\mathbf{P}^{-1} = \sum_i \lambda_i \mathbf{c}_i \mathbf{c}_i^\top = \mathbf{C}\mathbf{\Lambda}\mathbf{C}^\top,$$

where $\mathbf{C} = [\boldsymbol{c}_1, \boldsymbol{c}_2 ... \boldsymbol{c}_n] = \mathbf{K}_{*f}\mathbf{K}^{-1}$ and $\mathbf{\Lambda}$ is a diagonal matrix of the values of λ_i. The summation in the expression for L, in (4), $\sum_i \lambda_i(1 - \mathbf{c}_i^\top \mathbf{P}\mathbf{c}_i)$ can be written as $\text{Tr}\left(\mathbf{\Lambda} - \mathbf{C}^\top \mathbf{P}\mathbf{C}\mathbf{\Lambda}\right)$. Substituting in our definition of $\mathbf{P}$, we can write the summation as:

$$\text{Tr}\left(\mathbf{\Lambda} - \mathbf{C}^\top (\mathbf{C}\mathbf{\Lambda}\mathbf{C}^\top)^{-1}\mathbf{C}\mathbf{\Lambda}\right) = \text{Tr}\left(\mathbf{\Lambda} - \mathbf{C}^\top \mathbf{C}^{-\top}\mathbf{\Lambda}^{-1}\mathbf{C}^{-1}\mathbf{C}\mathbf{\Lambda}\right).$$

1. The summation is of a list of positive semi-definite rank-one matrices. One can see that such a sum is also positive semi-definite (to see this, consider distributing $\boldsymbol{z}^\top$ and $\boldsymbol{z}$ over the summation).

Assuming $\mathbf{C}$ is invertible, the summation becomes $\mathrm{Tr}\left(\mathbf{\Lambda} - \mathbf{\Lambda}^{-1}\mathbf{\Lambda}\right)$. Differentiating this wrt λ_j equals one. We can use this result to find the gradient of L wrt λ_j:

$$\begin{aligned}\frac{\partial L}{\partial \lambda_j} &= \frac{\partial \ln|\mathbf{P}|}{\partial \lambda_j} + \frac{\partial}{\partial \lambda_j}\sum_i \lambda_i(1 - \mathbf{c}_i^\top \mathbf{P}\mathbf{c}_i) \\ &= \mathrm{Tr}\left(\mathbf{P}^{-1}\frac{\partial \mathbf{P}}{\partial \lambda_j}\right) + 1 \\ &= \mathrm{Tr}\left(-\mathbf{P}^{-1}\mathbf{P}\boldsymbol{c}_j\boldsymbol{c}_j^\top\mathbf{P}\right) + 1 \\ &= -\mathrm{Tr}\left(\boldsymbol{c}_j\boldsymbol{c}_j^\top\mathbf{M}^{-1}\right) + 1 \\ &= -\boldsymbol{c}_j^\top\mathbf{M}^{-1}\boldsymbol{c}_j + 1.\end{aligned}$$

This can be solved using a gradient ascent method, to give us those values of λ_i which minimise $\log|\mathbf{M}|$ while ensuring $\Delta \leq 1$, repeating these steps,
Step 1: Compute,

$$\mathbf{M} = \sum_i \lambda_i \mathbf{c}_i \mathbf{c}_i^\top.$$

Step 2: Then update $\boldsymbol{\lambda}$ to maximise L, using,

$$\frac{\partial L}{\partial \lambda_j} = -\boldsymbol{c}_j^\top\mathbf{M}^{-1}\boldsymbol{c}_j + 1. \tag{7}$$

2.4 Differential Privacy and non-parametric methods

It is largely assumed that regression 'involves solving an optimization problem' (Zhang et al., 2012) and as discussed previously (Smith et al., 2018) there is little work currently proposing methods for DP non-parametric regression. We believe that, by avoiding a parametric formulation, we can also largely avoid the problem of iterative optimisation which the majority of parametric methods require. Similarly most examples for DP classification (e.g. Chaudhuri et al., 2011; Rubinstein et al., 2012; Song et al., 2013) work by perturbing the objective function and constraining the DP noise scale by carefully crafting the loss-function. In this paper we build on the cloaking method from Smith et al. (2018), a non-parametric (GP based) method for DP regression, as an alternative route to perform DP machine learning. Note that all the papers (besides Smith et al., 2018) consider the problem of making both the inputs and outputs of the training data private, thus aren't directly comparable to the cloaking method or its improvements outlined in this paper, which only consider output privacy.

The most common non-parametric DP regression method is the histogram, used as a baseline in Smith et al. (2018) and is described in Lei (2011). In both of these papers a non-DP regression algorithm is fitted to DP corrupted results of the histogram. When compared, the cloaking method of Smith et al. (2018) required less noise to achieve DP than using this intermediate histogram, although the histogram method is easy to extend to include privacy on the inputs, something absent in the cloaking method. Both the histogram and cloaking method are limited to low-dimensional input domains; the histogram method due to the

exponential number of bins that are required as dimensionality grows (and associated low-occupancy); the cloaking method due to excessive sensitivity to outliers. This paper proposes an alteration to the cloaking method to allow higher-dimensional training data.

DP non-parametric *classification* also has relatively little work associated. Although not truly non-parametric, Li et al. (2014) develop a protocol for computing the SVM's (linear) decision boundary normal vector in a DP manner. DP collaborative-filtering kNN methods have similarities and are non-parametric, and can be considered to be solving a classification problem. Examples include Zhu et al. (2014) and Afsharinejad and Hurley (2018). In the former, the selection of neighbours is made DP using the exponential mechanism. In the latter the training points are perturbed first (by corrupt encoding them) before performing a nearest neighbour operation. Then, in both, additional perturbation is conducted to protect those neighbours selected.

DP parametric classification methods are more common. Unlike this paper's aims, the work in this field has focused on classifiers without strong priors, for example using deep neural networks (Abadi et al., 2016), decision trees (Jagannathan et al., 2009; Fletcher and Islam, 2017; Li et al., 2020), or using a DP Bayes classifier (Vaidya et al., 2013). These and other results, e.g. Chaudhuri et al. (2011) provide algorithms which ensure the training *inputs* remain private too. Both of these aspects (input privacy and weak priors) mean that larger numbers of training points have been required. For example Abadi et al. (2016) used 50,000-60,000 training points; Chaudhuri et al. (2011) used 47,000 and 5 million; Zhang et al. (2012) used 190,000 and 370,000 (with linear and logistic regression); Afsharinejad and Hurley (2018) used over 100,000 (kNN). The importance of strong priors is a point emphasised by Dimitrakakis et al. (2017) 'robustness and privacy are linked via smoothness. Learning algorithms that are smooth mappings...are robust.' They explain that this robustness is provided by the prior: 'informative priors achieve better privacy, as the posterior has a weaker dependency on the data.'

So to emphasise, this paper addresses questions and problems of a fundamentally different nature to many of the papers in differentially private machine learning. First, we are interested in the problem of privacy on just the *outputs*. Second, our method targets the problem of learning from small data sets, unlike most of the above cited papers. In all our examples N$<$300. Informally, Gaussian process regression can achieve this due to the prior leading to smoothness.

3. Nonstationary Kernels

To quickly reiterate the problem. We have a set of training data, consisting of public inputs, $\mathbf{X}$, and private outputs, $\boldsymbol{y}$. We wish to make a prediction for a hypothesised latent function at location $\boldsymbol{x}_*$, using this training data, in such a way that an attacker will not gain much information about any individual value in $\boldsymbol{y}$.

Using the standard cloaking method from Smith et al. (2018), DP noise becomes unreasonably large in the regions *neighbouring* the data concentrations. Figure 2 illustrates this using a synthetic one-dimensional data set of ten data points. A GP regression has been performed and the posterior mean (and standard errors) have been drawn (red lines). The outlier datapoint is reduced from 0.4 to 0.15 (blue cross) leading to a very different posterior in the neighbouring part of the domain (perturbed in this example by up to six times the

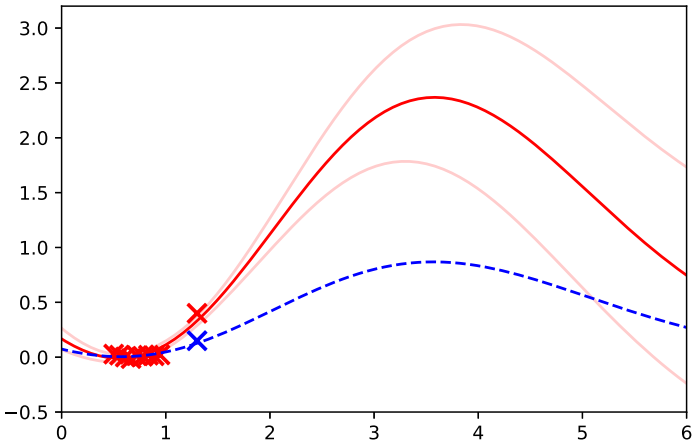

Figure 2: Why regions of the domain *neighbouring* the data need more DP noise. Synthetic data set marked with red crosses. Red lines show the GP regression posterior mean and standard deviation, using an EQ kernel (lengthscale=2, scale=1, $\sigma^2 = 0.001$). The outlier's output is perturbed by 0.25, leading to a large change in the posterior mean (dashed blue line) far from where the data lies.

change in the outlier). One can think of this sensitivity of the neighbouring posterior mean to outliers as due to a pivoting effect with the bulk of the data acting as the fulcrum around which the posterior mean function is pivoting. Figure 3A shows how this leads to high DP noise scales in the surrounding region for a real example. Figure 5A demonstrates how this problem worsens as the number of dimensions increases. With additional dimensions each data point has more opportunities to become an outlier. In much of the domain only a small number of training points contribute towards the prediction (if using a local kernel). Far more noise is then required to protect the individual training points.

We propose that this effect can be somewhat mitigated by manipulating the aggregation performed by the Gaussian process. Consider the structure of $\mathbf{C}$, from (1), this simply describes the amount each training point contributes to a prediction. If we imagine there are n outlying training points close together next to our test point then, for a kernel like the EQ, we will discover that their associated columns in $\mathbf{C}$ will all be approximately equal to $1/n$. Effectively they will average the training outputs. The more training data that is within this clique, the further the contribution provided by each training point is reduced. Obviously real data will lead to more complicated covariance behaviour, but a similar effect occurs—if more data lies within one-lengthscale of test point, then the effect of individual training points is diminished.

We initially consider using a non-stationary lengthscale to maintain approximately similar numbers of training points in the aggregation, effectively controlling for the density of training data at the test point location. We look at why this approach fails to achieve the desired aggregation.

As an alternative we consider the use of a sparse approximation using inducing inputs, placed over regions of high training density, through which the predictions are mediated. This has the practical effect of diminishing the contribution from those points far from the inducing inputs (i.e. the outliers).

Both of these methods can be considered a form of non-stationary kernel, as they both manipulate the influence of the training points in a way that varies depending on location.

3.1 Non-stationary lengthscale

One possible method for limiting the sensitivity of the posterior mean is through the manipulation of the contribution of training data to the prediction by using a non-stationary lengthscale, adjusted depending on the density of the data. With the intuition that if one uses longer lengthscales in regions with scarce data, the posterior mean will still be able to depend on many training points.

Here we are interested in using lengthscales that are effectively fixed by the density of the training data's input locations. We base our method on Gibbs (1998, p48) which allows one to produce a positive definite covariance matrix, if one already has *a priori* beliefs about the lengthscale at each location. We also found Paciorek and Schervish (2004) was helpful for extending to multiple input dimensions.

We briefly sketch how Gibbs (1998) constructs the non-stationary covariance function. Consider using B basis functions to model $y_n = \sum_{i=1}^{B} w_i \phi_i(\boldsymbol{x}_n) + \nu_n$ over data ($\mathbf{X} \in \mathbb{R}^{N\times L}$, $\mathbf{y} \in \mathbb{R}^N$), with design matrix built of evaluated basis functions, $\Phi \in \mathbb{R}^{N\times B}$. Using independent Gaussian priors (with scale parameters α and β on the scale of its weights $\mathbf{w}$ and noise terms respectively) the posterior has a predictive distribution proportional to,

$$p(\boldsymbol{y}|\mathbf{x}, \alpha, \beta) \propto \exp\left(-\frac{1}{2}\mathbf{y}^\top \mathbf{G}^{-1}\mathbf{y}\right)$$

where $\mathbf{G} = \alpha^{-1}\Phi\Phi^\top + \beta^{-1}\mathbf{I}$. One can see the similarity with the Gaussian process model, where $\mathbf{G}$ takes the place of the covariance matrix. To be more concrete, if one uses a set of Exponentiated Quadratic (EQ) basis functions at locations specified in $\mathbf{A}$,

$$\phi_k(\boldsymbol{x}) = \exp\left(-\sum_{l=1}^{L}\frac{(x^{(l)} - a_k^{(l)})^2}{r_l^2}\right),$$

and substitutes this into the outer product, and takes the limit of the number of bases, one is left with the familiar EQ covariance function,

$$G(\boldsymbol{x}_i, \boldsymbol{x}_j) \propto \exp\left(-\sum_{l=1}^{L}\frac{(x_i^{(l)} - x_j^{(l)})^2}{r_l^2}\right).$$

Gibbs (1998) then develops this by considering the EQ basis function with parameterised lengthscale (or radius), $r_l(\boldsymbol{x}; \Theta_l)$,

$$\phi_k(\boldsymbol{x}) = \prod_{l=1}^{L}\sqrt{\frac{\sqrt{2}}{r_l(\boldsymbol{x};\Theta_l)}}\exp\left[-\sum_{l=1}^{L}\frac{(x^{(l)} - a_k^{(l)})^2}{r_l^2(\boldsymbol{x};\Theta_l)}\right].$$

The prefactor here ensures that the resulting covariance function has a constant variance (wrt $\boldsymbol{x}$). After a rather involved derivation he shows that this leads to the covariance,

$$C(\boldsymbol{x}_m, \boldsymbol{x}_n; \Theta) = \theta_1 \prod_l \left\{ \frac{2 r_l(\boldsymbol{x}_m;\Theta) r_l(\boldsymbol{x}_n;\Theta)}{r_l^2(\boldsymbol{x}_m;\Theta) + r_l^2(\boldsymbol{x}_n;\Theta)} \right\}^{1/2} \times \exp\left(- \sum_l \frac{\left(x_m^{(l)} - x_n^{(l)}\right)^2}{r_l^2(\boldsymbol{x}_m;\Theta) + r_l^2(\boldsymbol{x}_n;\Theta)} \right). \quad (8)$$

We now need to select a lengthscale function, which gives the lengthscale for a given location; $r_l(\boldsymbol{x})$. If we knew the density, $\rho(\boldsymbol{x})$, of data at location $\boldsymbol{x}$ then we would want the lengthscale to be roughly $r_l(\boldsymbol{x}) = n/\rho(\boldsymbol{x})$, where n is the number of training points we want within half a lengthscale (in 1d) of $\boldsymbol{x}$. To avoid arbitrarily large lengthscales, we rearrange and add a small constant m^{-1} to the denominator, where m then is an upper bound on the lengthscale; $r_l(\boldsymbol{x}) = [m^{-1} + \rho(\boldsymbol{x})/n]^{-1}$. Finally, we approximate $\rho(\boldsymbol{x})$ using Kernel Density Estimation (KDE) and an exponentiated quadratic (EQ) kernel. As $\mathbf{X}$ is public these calculations do not involve differential privacy.

We will see in the results section that we found this initial design functioned poorly as long lengthscale areas were effectively made somewhat independent from the high-density short lengthscale areas. In an attempt to ameliorate this we modified the function further during experimentation (see Section 6.1) by using the smallest value of $\rho(\boldsymbol{x})$ from a region neighbouring $\boldsymbol{x}$ in an attempt to allow higher density regions to support the low-density areas.

3.2 Sparse Gaussian Processes

An alternative way to manipulate the kernel over the input space is through the use of a sparse, inducing point approximation. First we will summarise the results of the Fully Independent Training Conditional (FITC) approximation (Snelson and Ghahramani, 2006), a simple but widely used approach, before discussing how the effect of outliers on our DP predictions can be diminished by pushing the regression through this sparse approximation.

FITC and other related methods were devised to reduce the computational complexity of Gaussian process regression. The problem lies with the cost of computing, storing and inverting the full $N \times N$ covariance matrix between all the training points. Rather than use all the training data to predict the test point a small number $M << N$ of pseudo datapoints are invented and placed such that they model the original training data as well as possible. To derive FITC first assume that M inducing points are at fixed locations $\mathbf{X}_u$. These have *latent* values $\bar{\boldsymbol{f}}$, which we use to perform standard GP regression to compute the (conditionally independent) observations,

$$p(y_n|\mathbf{X}, \mathbf{X}_u, \bar{\boldsymbol{f}}) = \mathcal{N}\left(y_n \mid \boldsymbol{k}_n^\top \mathbf{K}_{MM}^{-1} \bar{\boldsymbol{f}}, K_{nn} - \boldsymbol{k}_n^\top \mathbf{K}_{MM}^{-1} \boldsymbol{k}_n + \sigma^2\right).$$

Where K_{MM} is the covariance between the inducing inputs and $\boldsymbol{k}_n$ is between the inducing inputs and training point n. Snelson and Ghahramani (2006) place the same GP prior on the inducing point values $\bar{\boldsymbol{f}}$, which allows them to integrate out $\bar{\boldsymbol{f}}$. This leaves a predictive

Gaussian distribution[2] with mean and variance,

$$\mu_* = \boldsymbol{k}_{*M}\mathbf{Q}_{MM}^{-1}\mathbf{K}_{MN}(\boldsymbol{\Lambda}+\sigma^2 I)^{-1}\boldsymbol{y} \tag{9}$$

$$\sigma_*^2 = K_{**} - \boldsymbol{k}_{*M}(\mathbf{K}_{MM}^{-1} - \mathbf{Q}_{MM}^{-1})\boldsymbol{k}_{M*} + \sigma^2.$$

Where, $\mathbf{Q}_{MM} = \mathbf{K}_{MM}+\mathbf{K}_{MN}(\boldsymbol{\Lambda}+\sigma^2 I)^{-1}\mathbf{K}_{NM}$, $\boldsymbol{\Lambda} = \text{diag}(\boldsymbol{\lambda})$ and $\lambda_n = K_{nn}-\boldsymbol{k}_{nM}\mathbf{K}_{MM}^{-1}\boldsymbol{k}_{Mn}$. $\mathbf{K}_{NM}$ is the covariance between training and inducing inputs, $\mathbf{k}_{nM}$ is the covariance between a given training point n and the inducing points. $\boldsymbol{k}_{*M}$ is the covariance between the test and inducing inputs and K_{nn} is the variance of training point n.

As before, the variance term only depends on the inputs ($\boldsymbol{y}$ does not appear in the expression for σ_*^2). Thus we only need to protect the expression for the mean. The cloaking matrix is now all the terms to the left of $\boldsymbol{y}$ in (9),

$$\mathbf{C} = \boldsymbol{k}_{*M}\mathbf{Q}_{MM}^{-1}\mathbf{K}_{MN}(\boldsymbol{\Lambda}+\sigma^2\mathbf{I})^{-1}.$$

We can again use the cloaking method, described in section 2 to find the optimal $\mathbf{M}$, using gradient descent, using (7).

By using inducing points we effectively down-weight training data that lies far from these inputs. Thus, by simply placing the inducing inputs at locations of high density, we are able to reduce the influence of the outliers, and thus reduce the noise required for privacy. This method allows us to continue to use our chosen covariance function and parameters. However it does require selecting the number and location of inducing inputs, and it introduces biases towards training data that lies at the most dense locations.

Often when using inducing points, their input locations are optimised to improve the model fit. However this requires looking at the outputs, and thus leaks private data. We find inducing points often end up over outliers and sections of the domain with little data, which may not be optimum for controlling the scale of the DP noise. To solve both these problems we place the inducing points using a k-means clustering algorithm.[3] This is often the method used to initialise normal inducing point input optimisation and only requires data from the inputs. Typically it will place the inducing points at those areas of high density.

Later, in Section 6.1, we will see the GP regression results with five inducing inputs placed using k-means clustering. The intuition is that these will then be placed sufficiently far from outliers that it is just the densest regions which will have much influence on the posterior mean.

4. Binary GP Classification

We next investigate methods to apply DP to GP classification. In GP classification one assumes that a latent function $f(\boldsymbol{x})$ exists which is 'squashed' through a link function to

2. The model remains a GP (with the same covariance function for training and test) if we ignore the covariance between test points (Quiñonero-Candela and Rasmussen, 2005).
3. A variety of methods exist for computing k-means. Typically the algorithm iteratively assigns points to cluster centroids (depending on which is closest) and then updates the centroid locations to the mean of the points it has been assigned.

give us a distribution over the class probabilities $\pi(\boldsymbol{x})$. However this leads to an analytically intractable integral over the likelihood. This is therefore approximated, either numerically, or using analytical approximations, such as the Laplace approximation method or expectation propagation. To incorporate DP we chose to use the Laplace approximation, as it requires very few iterations for convergence and has a clear and simple relationship between the training outputs and the update step's direction, which is also of a form amenable to the application of the cloaking method.

From Williams and Rasmussen (2006), to apply the Laplace approximation we replace the exact posterior distribution $p(\boldsymbol{f}|\mathbf{X}, \boldsymbol{y})$ with an approximation $q(\boldsymbol{f}|\mathbf{X}, \boldsymbol{y}) = \mathcal{N}(\boldsymbol{f}|\hat{\boldsymbol{f}}, \mathbf{A}^{-1})$, where $\hat{\boldsymbol{f}}$ is the mode of p, and $\mathbf{A}$ is the Hessian of the negative log posterior at the mode. There is no analytical expression to find the mode. Instead, an algorithm is used in which an initial estimate for $\hat{\boldsymbol{f}}$ is iteratively updated,

$$\hat{\boldsymbol{f}}^{new} = (\mathbf{K}^{-1} + \mathbf{W})^{-1}(\mathbf{W}\hat{\boldsymbol{f}} + \nabla \log p(\boldsymbol{y}|\hat{\boldsymbol{f}})), \tag{10}$$

where $\mathbf{K}$ is the covariance between training input points and $\mathbf{W}$ is a diagonal matrix; $\mathbf{W} = -\nabla\nabla \log p(\boldsymbol{y}|\boldsymbol{f})$. The elements of $\mathbf{W}$ for the logistic link function equal $-\pi_i(1-\pi_i)$ where $[\pi(\hat{\boldsymbol{f}})]_i = \pi_i = p(y_i = 1|f_i) = (1 + e^{-f_i})^{-1}$. To make this update differentially private we need to consider how the training outputs $\boldsymbol{y}$ alter $\hat{\boldsymbol{f}}^{new}$. If we are using the logistic link function, the gradient term $[\nabla \log p(\boldsymbol{y}|\hat{\boldsymbol{f}}))]_i = t_i - \pi_i$, where $\boldsymbol{t} = \boldsymbol{y}/2 + \mathbf{1}/2$, so we can replace occurrences of $\nabla \log p(\boldsymbol{y}|\hat{\boldsymbol{f}}))$ with $\boldsymbol{y}/2 + \mathbf{1}/2 - \pi(\hat{\boldsymbol{f}})$.

We define

$$\mathbf{C} = \frac{1}{2}(\mathbf{K}^{-1} + \mathbf{W})^{-1}. \tag{11}$$

Substituting in this definition and the gradient term replacement into (10) we can rewrite the update as,

$$\hat{\boldsymbol{f}}^{new} = 2\mathbf{C}(\mathbf{W}\hat{\boldsymbol{f}} + \boldsymbol{y}/2 + \mathbf{1}/2 - \pi(\hat{\boldsymbol{f}})) = 2\mathbf{C}(\mathbf{W}\hat{\boldsymbol{f}} + \mathbf{1}/2 - \pi(\hat{\boldsymbol{f}})) + \mathbf{C}\boldsymbol{y}.$$

Here the last term is the only one that depends on $\boldsymbol{y}$ (both $\mathbf{K}$ and $\mathbf{W}$ are independent of $\boldsymbol{y}$). Thus we only need to consider the effect of changes in $\boldsymbol{y}$ on $\mathbf{C}\boldsymbol{y}$. We can use the *cloaking method*, described in Section 2.3, with a cloaking matrix as defined in (11). This allows us to estimate $\hat{\boldsymbol{f}}$ in a DP manner.

To make a prediction we need to compute the mean and variance of the latent function at a test point $\boldsymbol{x}_*$. We compute $\boldsymbol{k}_*$, the covariances between this test point and the training inputs. In Williams and Rasmussen (2006) the value of the latent function's mean at a test point is computed as $\mathbb{E}_q[f_*] = \boldsymbol{k}_*^\top \nabla \log p(\boldsymbol{y}|\hat{\boldsymbol{f}})$. However, the gradient term depends on the values of $\boldsymbol{y}$. To avoid this additional leakage of privacy, we instead use the approximation $\nabla \log p(\boldsymbol{y}|\hat{\boldsymbol{f}}) = K^{-1}\hat{\boldsymbol{f}}$ (Williams and Rasmussen, 2006, eq 3.21. At the mode the gradient is zero and so eq. 3.13 is zero), for which we have already used our privacy budget. So the mean of the posterior $\mathbb{E}_q[f_*] = \boldsymbol{k}_*^\top \mathbf{K}^{-1}\hat{\boldsymbol{f}}$, and the variance $\mathbb{V}_q[f_*] = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{k}_*^\top(\mathbf{K} + \mathbf{W}^{-1})^{-1}\boldsymbol{k}_*$. We have found that $\mathbf{K}$'s inverse can contain very large positive and negative values, particularly if training points are highly correlated (and thus approach a singular matrix). However, entries in $\hat{\boldsymbol{f}}$, associated with these correlated training points will have similar values due to the covariance structure in $\mathbf{C}$ and so will largely cancel the large positive and negative values in $\mathbf{K}$'s inverse. As explained in Section 2.3, if there are more

training points in a particular area around a point i, the DP cloaking approximation to $[\mathbf{C}\boldsymbol{y}]_i$ for that element will be more accurate than for elements associated with outliers.

Note that for numerical stability, we use several alternative but equivalent expressions, as recommended in Williams and Rasmussen (2006, pages 45-47). Specifically (defining $\mathbf{B} = \mathbf{K} + \mathbf{W}^{1/2}\mathbf{K}\mathbf{W}^{1/2}$) we compute $\mathbf{C} = \frac{1}{2}(\mathbf{K} - \mathbf{K}(\mathbf{W}^{1/2}\mathbf{B}^{-1}\mathbf{W}^{1/2}\mathbf{K}))$. See reference for additional tricks to achieve numerical stability and computational efficiency.

4.1 Sparse GP Classification

We can modify the above method to incorporate a sparse approximation similar to the one introduced in section 3.2, but this time we simply substitute a low-rank approximation for $\mathbf{K}$, following the Subset of Regressors (SoR) approximation (Quiñonero-Candela and Rasmussen, 2005; Smola and Bartlett, 2001), which corresponds to replacing the covariance function with,

$$\mathbf{K} = \mathbf{K}_{NM}\mathbf{K}_{MM}^{-1}\mathbf{K}_{NM}^{\top}.$$

Where $\mathbf{K}_{NM}$ is the covariance between the N training inputs and M inducing inputs; $\mathbf{K}_{MM}$ is the covariance between inducing inputs. Quiñonero-Candela and Rasmussen (2005) note that the SoR prior can cause degeneracy in the prediction of the posterior variance, but we are only interested in the posterior *mean* for the purposes of this component, as it is only the mean that is affected by the value of the training data's outputs. We could have used the FITC method, mentioned earlier, by adding $\boldsymbol{\Lambda}$ to the above expression, which would give a slightly better approximation, but for simplicity we felt the SoR approximation was sufficient.

We place the M inducing points using k-means clustering, as before, as this only depends on the training inputs (which are not private in this scenario). The posterior covariance can also be computed using one of the sparse methods, if required, in a non-DP manner.

5. Hyperparameter Selection

So far in this paper we have selected the values of the kernel hyperparameters *a priori*. Normally, in the non-DP setting, one may maximise the marginal likelihood to select these values or potentially integrate over the hyperparameters. In differential privacy we must take care when using private data to make these choices. Previous work exists to perform this selection, for example Kusner et al. (2015) describes a method for performing differentially private Bayesian optimisation, however their method assumes the training data is not private. Kusner et al. (2015) do suggest that the work of Chaudhuri and Vinterbo (2013) may allow Bayesian optimisation to work in the situation in which the training data also needs to be private, we outline here a more direct solution.

Chaudhuri et al. (2011) proposed a tuning algorithm for selecting a classifier's (hyper)parameters by using the exponential mechanism. This maximises the probability of selecting the optimum hyperparameters while still ensuring DP. With modification this could apply to selecting hyperparameters for *regression*. The bounding of the sensitivity requires a little more thought than in the classification case though.

Due to the low-dimensionality of many hyperparameter problems, a simple grid search, combined with the exponential mechanism may allow the selection of an acceptable set of hyperparameters. To summarise, from Dwork and Roth (2014), the exponential mechanism is a method for selecting in a DP way the optimal (highest) choice from a set of utilities, in such a way that the values being selected over remain private. One must provide the sensitivity of the utility (i.e. an upper bound on how much a training point can change the utility). Using the utilities and their sensitivities, the exponential mechanism selects each item with a given probability, such that the items with the greatest utility are most likely to be selected. The selection will not always select the actual maximum option, but it can be shown that the mechanism performs optimally while still remaining differentially private. Quoting Dwork and Roth (2014, definition 3.4),

> The exponential mechanism $M_E(x, u, \mathcal{R})$ selects and outputs an element $r \in \mathcal{R}$ with probability proportional to,
>
> $$p_r \propto \exp\left(\frac{\varepsilon u(x, r)}{2\Delta u}\right). \tag{12}$$

Where, x is the database, r is one of a finite number of possible outputs: $r \in \mathcal{R}$. u is the utility function. To use the exponential mechanism one evaluates the utility $\mathrm{u}(\{\boldsymbol{f}_*, \boldsymbol{y}_t\}, \theta_i)$ for a given database (which in our case consists of predictions and test outputs $\{\boldsymbol{f}_*, \boldsymbol{y}_t\}$) and for element θ_i, from a set Θ (which consists of all the hyperparameter configurations we wish to compare). One also computes the sensitivity, Δ_u, of this utility function. Then one selects an element θ_i with probability proportional to the expression in (12). Quoting Dwork and Roth (2014) again, 'we care only about the sensitivity of u with respect to its database argument; it can be arbitrarily sensitive in its range argument.' In our case this means,

$$\Delta_u = \max_{\theta_i \in \Theta} \max_{x \sim y} |u(x, \theta_i) - u(y, \theta_i)|,$$

where databases x and y differ by one row. In our case this difference represents a change in one of the values of $\boldsymbol{y}$. The rest of this section looks at a method for computing this utility and its sensitivity.

Usually when optimising hyperparameters for GPs one might use the marginal log likelihood (MLL).[4] However the additional DP noise means the maximum likelihood solution might not give the most accurate answers, as it will be optimising for the best non-DP solution, while we want to produce predictions that are the most accurate while preserving privacy. The contribution of the DP noise will need to be included in the estimate of output accuracy. So, for the utility function, for simplicity in estimating the sensitivity bound, and to avoid overfitting, we used the sum square error (SSE). The SSE is computed over a series of κ-fold cross-validation runs, which for the N_k points in the kth fold is

$$\mathrm{SSE}_k = \sum_{i=1}^{N_k} \left(f_{*i} - y_{\mathrm{t}i}\right)^2,$$

4. This method effectively penalises more complex models as they require integrating over a larger domain of parameters, leading to a balance between data fit and model complexity. See page 113 of Williams and Rasmussen (2006) for a discussion.

where the two terms are elements from the predictions $\boldsymbol{f}_*$ and test values $\boldsymbol{y}_\mathrm{t}$, respectively. This is somewhat similar to the method described in Chaudhuri et al. (2011), but we need further thought regarding the sensitivity of the utility function. In Chaudhuri et al. (2011) this was simply equal to 1 as the utility was equal to the number of misclassifications.

The choice of splits for the cross-validation is arbitrary, as long as there is no dependence on the private data. In particular there is no need to find the 'worse case' split. The sensitivity of a given choice of split is only dependent on the inputs (which are public). In theory one could select the cross-validation splits to minimise the resulting sensitivity, however in practice this might also end up compromising the choice of hyperparameter, as those splits might not reflect the expected structure of training and test data when applied. For example a split in which the test data is very evenly distributed would need less DP noise added, and so a shorter lengthscale would be supported than in the likely application.

Before proceeding we need to compute a bound on the sensitivity of the SSE. To briefly recap, the DP assumption is that one data point has been perturbed by at most d. We need to bound the effect of this perturbation on the SSE. First we realise that this data point will only be in the *test set* in one of the κ folds. In the remaining folds it will be in the training data.

If the perturbed data point is in the training data ($\boldsymbol{y}$), then we can compute the sensitivity of the SSE using the cloaking mechanism previously described. The perturbation this would cause to the predictions ($\boldsymbol{f}_*$) is described using standard GP regression (and the cloaking matrix). Specifically, $\boldsymbol{f}_* = \mathbf{C}\boldsymbol{y}$. So a change of d in training point j will cause a $d\boldsymbol{c}_{jk}$ change in the test point predictions, where $\boldsymbol{c}_{jk}$ is the jth column of the cloaking matrix for the kth fold.

To compute the perturbation in the SSE caused by the change in the training data, we note that the SSE is effectively the square of the Euclidean distance between the prediction and the test data. We are moving the prediction by $d\boldsymbol{c}_{jk}$. The largest effect that this movement of the prediction point could have on the distance between prediction and test locations is if it moves the prediction in the opposite direction to the test points. Thus it can increase (or decrease) the distance between the test and predictions by the largest length of $d\boldsymbol{c}_{jk}$ from all training points for fold k. Hence for one of the folds, the largest change in the SSE is $d^2 \max_j |\boldsymbol{c}_{jk}|_2^2$.

If the perturbed data point, j, was in the test data then the SSE will change by $(f_{*j} + d - y_{\mathrm{t}j})^2 - (f_{*j} - y_{\mathrm{t}j})^2 = d^2 + 2d(f_{*j} - y_{\mathrm{t}j})$. The last part of the expression (the error in the prediction for point j) is unbounded. To allow us to constrain the sensitivity we enforce a completely arbitrary bound of $|f_{*j} - y_{\mathrm{t}j}| \leq 4d$, thresholding the value if it exceeds this. Thus a bound on the effect of the perturbation if it is in the test set is $d^2 + 2d \times 4d = d^2 + 8d^2 = 9d^2$. The consequence of restricting this error is that we will underestimate the SSE when selecting a hyperparameter configuration, but this will only affect configurations with particularly high SSE, and thus will be those with very low probability anyway. In the algorithm below we effectively threshold both $|f_{*j} - y_{\mathrm{t}j}|$ and $|f_{*j} + d - y_{\mathrm{t}j}|$ which could lead to a potentially tighter bound.

The SSE of each fold is added together to give an overall SSE for the cross-validation exercise. We sum the $\kappa - 1$ largest training case sensitivities and add $9d^2$ to account for the effect of the single fold in which the perturbing data point, j, will be in the test set. The perturbation could have been in the test data in any of the folds. The effect of the pertur-

bation would be maximised if it was in the fold with the smallest training-data sensitivity. If it had been in any other fold the sensitivity would have been less. Thus the sensitivity (for hyperparameter configuration $\theta_i \in \Theta$) of the SSE over the κ folds is

$$\Delta_u^{(\theta_i)} = 9d^2 + \sum_{k=1}^{\kappa-1} d^2 \max_j |\boldsymbol{c}_{jk}|_2^2, \tag{13}$$

where the κ folds are ordered by decreasing sensitivity (so that $k = 1$ has the greatest, etc). The sensitivity for the exponential mechanism is the largest of all the possible configurations,

$$\Delta_u \triangleq \max_{\theta_i \in \Theta} \Delta_u^{(\theta_i)}. \tag{14}$$

Thus, we compute the SSE and the SSE's sensitivity for each of the hyperparameter combinations we want to test and select the largest sensitivity from each combination to use as Δ_u. We then use the computed sensitivity bound with the exponential mechanism to select the hyperparameters. So to reiterate, in this case each utility corresponds to a negative SSE, and the sensitivity, Δ_u, corresponds to the maximum (14). Equation (12) defines the probabilities we use to select the hyperparameters θ_i using the mechanism.

Note that for a given privacy budget, some ε will need to be expended on this selection problem, and the rest expended on the actual regression.

A significant problem with the naïve application of the above method is that the sensitivity, Δ_u, is dominated by hyperparameter configurations which have the greatest sensitivity but are unlikely to have high utility. This leads to the probabilities of each configuration becoming almost equal. The sensitivity calculation depends on just $\boldsymbol{c}_{jk}$ and the specified data sensitivity, d. Thus it doesn't depend on the private outputs and therefore we can selectively ignore those configurations with a sensitivity above a certain threshold. This might seem to be potentially discarding an optimum configuration, however configurations in which the SSE is highly sensitive to a single training point are also those in which the predictions themselves are sensitive requiring destructive levels of DP noise. So it is likely they will not be the configurations that achieve the best SSE.

Finally, when estimating the SSE for each configuration we include the effect of the DP noise (specifically we sample many times from the DP noise distribution) to ensure that this is included in the final SSE. The noise is independent of private data so this addition doesn't introduce extra DP requirements. This is so that when selecting the configuration we take into account the cost of the DP noise. The full calculation is described in Algorithm 1.

The algorithm could be further improved with some simple modifications. Note, for example, that the same point has been changed (in the same way) across all the cross validation splits. One could therefore find the training point that maximises the sum in (13), rather than sum the maximums. Given it is the same training point that we assume is perturbed, this is more efficient than finding the maximum training point for each fold. One could express this by constructing each fold's $\mathbf{C}$ matrix with the same indices for each data point, and set to zero those entries which are in the test set in the cross-validation fold. One would then, for each j, sum over $|\mathbf{c}_{jk}|_2^2$ for *all* κ folds, and find the maximum. Improvements can also be made to the underlying exponential algorithm, for example by replacing it with more recent approaches, such as randomised early stopping (Liu and Talwar, 2019) which are likely to achieve better utility for the same DP budget.

Algorithm 1 Hyperparameter selection using the exponential mechanism.

Require: $\mathcal{M}$ and Θ; the GP model and the hyperparameter configurations we will test.
Require: $\mathbf{X}_{\text{all}} \in \mathbb{R}^{N\times D}, \boldsymbol{y}_{\text{all}} \in \mathbb{R}^{N}$; inputs and outputs.
Require: $d > 0$, data sensitivity & $\varepsilon > 0, \delta > 0$, the DP parameters.
Require: $\kappa > 1$, the number of folds in the cross-validation.

function HyperparameterSelection($\mathbf{X}_{\text{all}}$, $\boldsymbol{y}_{\text{all}}$, $\mathcal{M}$, Θ, d, ε, δ, κ)
  for $\theta \in \Theta$ **do**
    $\text{SSE}^{(\theta)} \leftarrow 0$
    for $k \in \kappa$ **do**
      Split $\mathbf{X}_{\text{all}}$, $\boldsymbol{y}_{\text{all}}$ into kth fold training ($\mathbf{X}$,$\mathbf{y}$) and $N_t^{(k)}$ test ($\mathbf{X}_t$,$\mathbf{y}_t$) points.
      $\mathbf{C}_k \leftarrow \mathcal{M}.\text{get\_C}(\mathbf{X}, \mathbf{X}_t, \theta)$ ▷ Compute cloaking matrix, e.g. $\mathbf{K}_{*f}\mathbf{K}^{-1}$ for kth fold.
      $\mathbf{M}, \Delta \leftarrow \mathcal{M}.\text{get\_M}(\mathbf{C}_k)$ ▷ Compute noise covariance terms and Δ.
      $\text{SSE}^{(\theta)} \leftarrow \text{SSE}^{(\theta)} + \sum_{i=1}^{N_t^{(k)}} \min\left(\max\left(f_{*i} - y_{\text{t}i}, -4d\right), 4d\right)^2$
      $\text{SSE}^{(\theta)} \leftarrow \text{SSE}^{(\theta)} + \sum_{i=1}^{N_t^{(k)}} \left(f_{*i} - y_{\text{t}i}\right)^2$ ▷ Add this fold's SSE to total.
      $\alpha_k \leftarrow \max_j |\boldsymbol{c}_{jk}|_2^2$
    end for
    Sort $\boldsymbol{\alpha}$ to be ascending
    $\Delta_u^{(\theta)} \leftarrow 9d^2 + \sum_{k=1}^{\kappa-1} d^2\alpha_k$
  end for
  Select hyperparameter config from Θ with probability proportional to, $\exp\left(\frac{-\varepsilon \text{SSE}^{(\theta)}}{2\max(\Delta_u^{(\theta)})}\right)$
end function

5.1 Linear Regression Example

We find empirically that the addition of DP noise in the calculation of the SSE leads to less complex models being chosen, as these typically have less sensitivity to individual training points: The model chosen is not necessarily reflective of the true underlying data generating process, but rather is the model that will provide the most accurate predictions given the additional DP noise. To illustrate this and the hyperparameter selection algorithm in action we consider a highly simplified example with four data points, $\boldsymbol{x}_{all} = [0, 1, 2, 4]$, $\boldsymbol{y}_{all} = [0, \frac{1}{2}, 1, 2]$, split for cross validation in half. We wish to use the hyperparameter selection algorithm to select between a 0th and 1st order polynomial. *Without* DP noise added, the SSE for these two models are respectively 6.875 (from the two folds, 3.625+3.25) and 0 (0+0).

We note that for training ($\mathbf{X}$) and test ($\mathbf{X}_*$) design matrices, and training data $\boldsymbol{y}$ the predictions at the test locations are,

$$\boldsymbol{f}_* = \mathbf{X}_*(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y},$$

from which we can see the cloaking matrix is $\mathbf{X}_*(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top$. For the first cross-validation split the cloaking matrices for the 0th order polynomial (i.e. just finding the mean), and 1st order polynomial (linear regression) are:

$$\mathbf{C}_0 = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \qquad \mathbf{C}_1 = \begin{bmatrix} -1 & 2 \\ -3 & 4 \end{bmatrix}.$$

With (1,0.01)-DP and a sensitivity of 2, applying the cloaking mechanism, the SSEs are 50 and 1597 respectively. We can see why the linear regression requires far more DP noise by considering how a perturbation in the 2nd output $y^{(2)}_{all}$ between 0 and 2 can cause the prediction of $y^{(4)}_{all}$ to vary between 0 and 8, while in the 0th order model such a perturbation will only cause the prediction (average) to vary by 1. The effect of this is for the utility (we use the negative SSE) to be far lower for the 1st order model. We compute the maximum sensitivity, $\Delta_u = 116$, using (13) and (14), this results in the mechanism selecting the 0th-order polynomial with probability 99.87%.

This toy example allows us to briefly consider what would have happened if we had chosen a different split, for example split between interleaved pairs. This results in a smaller SSE for the 1st-order polynomial (337) and a smaller sensitivity $\Delta_u = 53$, leading to a probability of selecting the 0th-order polynomial of 93.94%. Both are valid results, and achieve DP, so it is not necessary to find the 'worst-case' cross-validation split. As usually is the case in ML it may be best to chose a split that reflects the likely distribution of training and test data in the application, as this will then make the added DP noise more representative of that likely to be experienced in use.

6. Results

We test the above methods with a series of experiments. Firstly, we investigate the non-stationary covariance methods, deployed to reduce outlier noise. We explored their effect in one and two dimensions in Sections 6.1 and 6.2 respectively. In Section 6.3 we look at

reasons for why varying the lengthscale doesn't reduce sensitivity to outliers. In Sections 6.4 and 6.5 we experiment with the binary DP classification method in one and two dimensions. In Section 6.6 e test the sparse DP classification method on the high dimensional MNIST data set. Finally, Section 6.7 demonstrates how DP (hyper)parameter selection might be performed.

6.1 Non-stationary covariance: 1d !Kung

In the examples below we experiment with both variable lengthscales and sparse approximations, as candidate methods for reducing the DP noise added in outlier regions.

We use, as a simple demonstration, the heights and ages of 287 women from a census of the !Kung (Howell, N., 1967). We are interested in protecting the privacy of their heights, but we are willing to release their ages. We used an EQ kernel with lengthscale set *a priori*, to 15 years as from our prior experience of human development this is the timescale over which gradients vary. Similarly we set the kernel variance to 10cm^2 and the Gaussian likelihood noise variance to 25cm^2. In the following we target (1,0.01)-DP unless stated otherwise.

Figure 3 demonstrates the three methods for reducing the DP noise added to the predictions. Figure 3C shows the effect of the sparse cloaking method in which five inducing inputs, placed using k-means clustering, have been used to reduce the sensitivity to outliers. For those unfamiliar with inducing point approximations it is worth noting that the posterior mean and covariance of the non-DP GP are only slightly changed by this approximation, which is expected given the lengthscale of 15 years, and the relative density of inducing points. Unlike the 'subset of data' (SoD) approximation (Quiñonero-Candela and Rasmussen, 2005) the FITC approximation achieves better uncertainty quantification, avoiding the wide distributions of SoD. Of importance to this paper is the marked reduction in DP noise in the outlier region beyond 70 years, with predictions now with some utility. Figure 4 demonstrates the benefit of the inducing point approximation, comparing the RMSE for both DP and non-DP regression using different numbers of inducing points. As we might expect both the DP and non-DP approaches perform poorly with very small numbers of inducing points. DP regression reaches its optimum with five or six inducing points. Here the data is well described by the sparse approximation (as we can see from the non-DP line) and thus additional inducing points will merely cause more DP noise without providing significant improvements to the approximation. Section 6.6 and Figure 8 investigate the same question for the MNIST data set, in particular the interaction with lengthscale.

In figure 3B, the variable lengthscale method is demonstrated. We found this method to perform poorly (see Table 1). Figure 3D shows the lengthscale used. Specifically we've constructed the lengthscale function specifically to try and mitigate the poor results, by adjusting the lengthscale such that regions next to outlier areas also have relatively long lengthscales. The intention being that these dense 'edge' regions can support predictions into the outlier areas. However we found we could only go so far, as increasing the lengthscale meant the GP would become unable to fit the true latent function's structure.

6.2 Non-stationary covariance: 2d !Kung

At higher dimensions more of the data becomes an outlier (consider how there are more directions in which a data point can remove itself from the manifold in which the majority

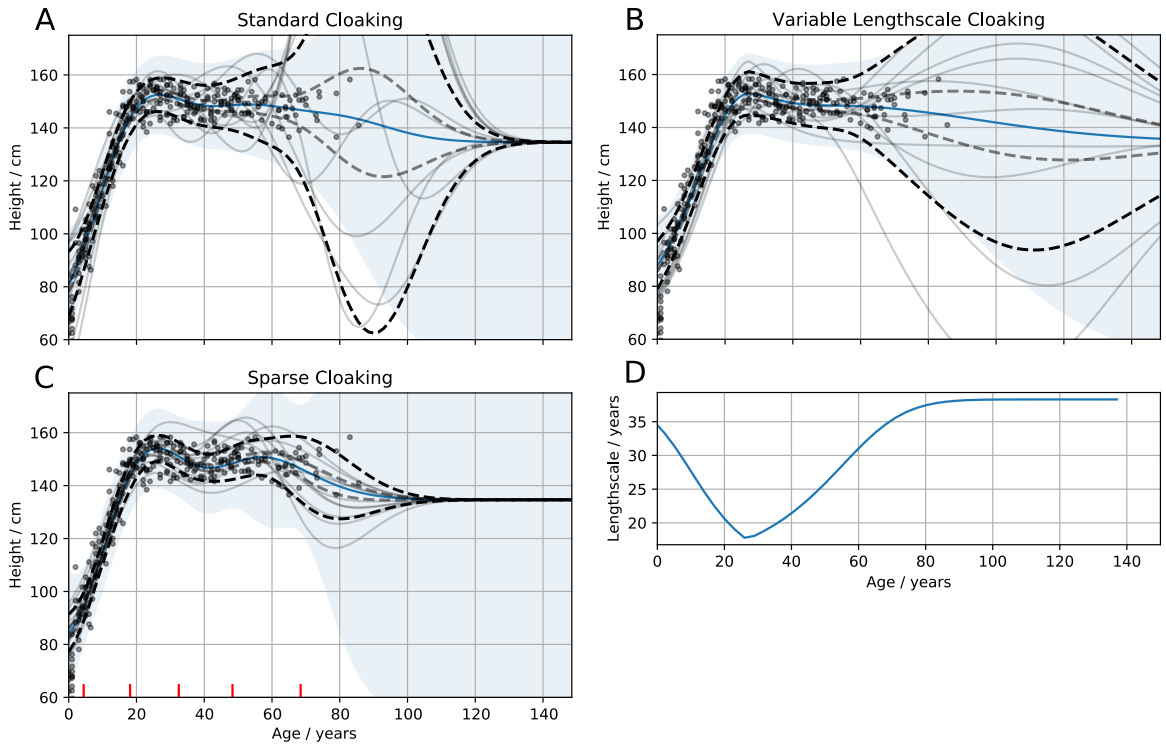

Figure 3: !Kung heights vs ages. In (A)-(C), solid blue lines, posterior non-DP means of the GPs; grey lines, DP samples; Black and grey dashed lines, standard error and $\frac{1}{4}$ standard error confidence intervals for DP noise respectively; blue area, non-DP GP posterior variance (excluding noise). $\varepsilon = 1$, $\delta = 0.01$, $\Delta = 100$ cm. (A) Standard cloaking method. Considerable DP noise added in region of outliers (70-110 years). (B) Cloaking with variable lengthscale GP. (C) Cloaking, using five inducing inputs (placed using k-means clustering). (D) The varying lengthscale used in (B).

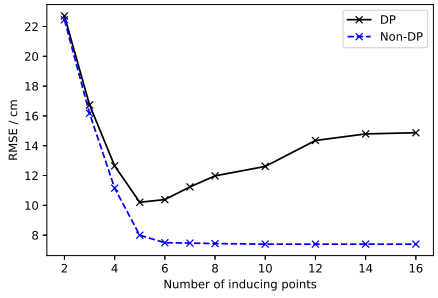

Figure 4: How the number of inducing inputs affects the accuracy on the 1d !Kung data set. Blue-dashed line: Non-DP accuracy. Black line: (1,0.01)-DP accuracy. 14-fold cross-validation. $l = 15$ years, kernel variance=10cm^2, $\sigma^2 = 25$cm^2.

	standard cloaking	sparse	nonstationary
1d	13.3 ± 3.2	9.9 ± 1.7	12.8 ± 2.0
2d	17.2 ± 3.1	10.2 ± 1.6	16.3 ± 2.5
1d (no DP)	7.1 ± 1.8	7.7 ± 1.9	8.6 ± 2.2
2d (no DP)	5.2 ± 1.5	7.7 ± 2.0	5.9 ± 1.7

Table 1: RMSEs estimating height (cm) from weight and age. Using 14-fold cross-validation runs on the !Kung data set. First two rows use $\varepsilon = 1.0$, $\delta = 0.01$, $\Delta = 100$cm. The second two rows have no DP noise added. Standard deviation over 14-folds indicated by $\pm$.

of the data lies). To demonstrate, we've added a new dimension (weight) to the !Kung data inputs (leaving the kernel as the exponentiated quadratic but now with two inputs). In figure 5 the most opaque discs are those will the least DP noise. In the top figure, we see that the majority of the domain has so much DP-noise added that the discs have been rendered completely transparent, with only a tiny part of the input space available for making predictions. For example the noise added for a test point at 50kg, aged 60, has a standard deviation of 40cm (approximately 25% of the non-DP prediction). Figures 5B and C illustrate the improvement in DP-noise when using the sparse method and the variable lengthscale method.

Importantly in both non-stationary methods the DP-noise at the prediction is much reduced. However the non-DP mean predictions differ, as these non-stationarities effectively define different models. In summary both non-stationary methods allow much more of the domain to be used to make predictions without DP destroying the prediction.

To further illustrate, we performed a cross-validation run on the two data sets. One would expect that providing the additional information about their weight into the model, would improve the estimates of their height. However for the standard cloaking method, the additional DP noise caused by the scarcity of data leads to an *increase* in the RMSE, while the sparse cloaking method is not only better in the 1d case, but is not degraded by the additional weight information. It is worth noting that the non-stationary methods were introduced to improve accuracy in outlier low-density areas, but the RMSE is over all data, most of which is marginally degraded by the approximation while the few outliers that are the beneficiaries of the approach, but contribute little to the RMSE metric.

In summary, using a sparse approximation with inducing points placed at locations of high data density, we are able to perform DP GP regression over the whole domain without excessive noise in outlier regions. The alternative variable-lengthscale strategy however, doesn't appear to achieve the same goal.

6.3 Notes on the variable lengthscale results

We briefly look at why the non-stationary kernel solution is not achieving the desired reduction in DP noise. Recall that when making predictions we want to average over sufficient training points so that individual training points only have a small effect on the prediction.

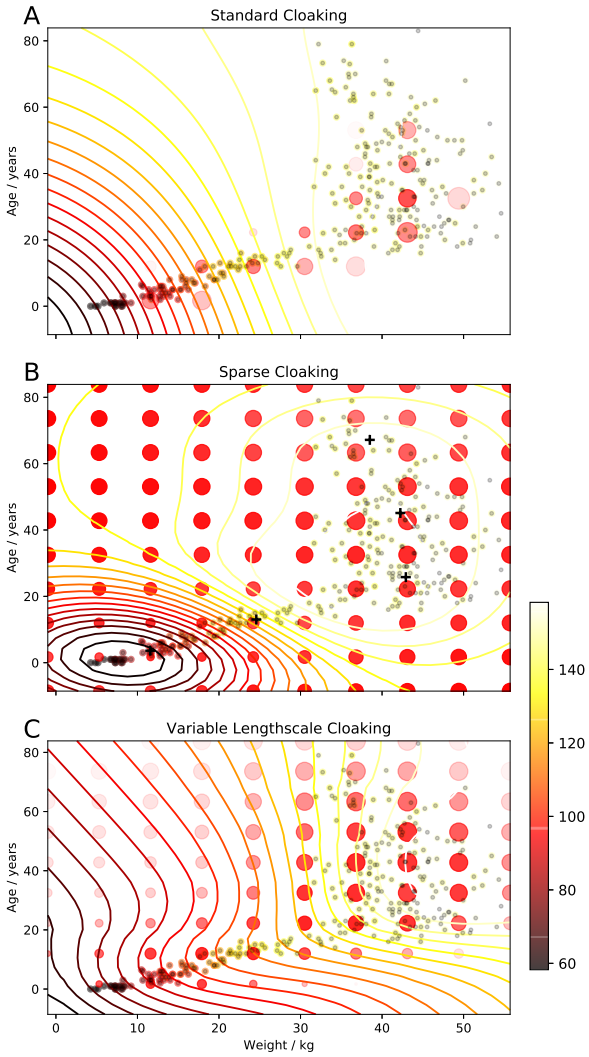

Figure 5: Predicting !Kung height from their weights and ages. Contours indicate non-DP posterior mean height predictions (in cm); dots, training inputs; disks, which are plotted in a grid over the whole domain, indicate DP test predictions (with size proportional to predicted height; transparency to DP-noise, with *full transparency reached when noise standard deviation reaches 40cm*). $\varepsilon = 1$, $\delta = 0.01$, $\Delta = 100$ cm. (A) Standard cloaking method. Only able to make low-noise predictions along part of a limited manifold (B) Sparse cloaking, using five inducing inputs (marked with crosses, placed using k-means clustering). DP noise reduced across whole domain. (C) Cloaking with variable lengthscale GP.

Thus we want, for a test point in the region of outliers, to still have a high covariance with many training points necessarily including some from high density regions. Conversely we want a test point in the middle of the high density region to only have a high covariance with nearby training data (i.e. short lengthscale) to allow it to describe more complex structure.

We can see, with a little thought, that these two requirements can't both be met. Consider a pair of points, A and B, within the short-lengthscale regime and a third point, C, in the long-lengthscale regime such that the three points are evenly spaced. We want, for predictions near C, a high covariance between the pairs A-C and B-C, but we also want a low covariance between the short-lengthscale pair A-B. We might also want a lower-covariance between A-C when predicting near A—the opposite of that required for the predictions at C.

This intuition is visible in the construction of the prefactor in (8): if the lengthscales for axis l, $r_l(\boldsymbol{x}_m;\Theta)$ and $r_l(\boldsymbol{x}_n;\Theta)$, associated with the two points, are similar then axis l's prefactor $\frac{2r_l(\boldsymbol{x}_m;\Theta)r_l(\boldsymbol{x}_n;\Theta)}{r_l^2(\boldsymbol{x}_m;\Theta)+r_l^2(\boldsymbol{x}_n;\Theta)}$ will be maximised. Of interest is the situation where the two lengthscales are very different. Even if the two inputs are close together the two distributions are poorly matched and thus their product will be small, leading to a low covariance: If we consider two nearby points (separated by distance d) and differentiate (8) with respect to the lengthscales of input $r_l(\boldsymbol{x}_m;\Theta)$, and set to zero. The choice of $r_l(\boldsymbol{x}_m;\Theta)$ which will maximise the covariance approaches the other input's covariance $r_l(\boldsymbol{x}_n;\Theta)$ in the limit as $r_l(\boldsymbol{x}_m;\Theta) >> d$. Simply put, location pairs with wildly different lengthscales will have a low covariance.

As an alternative we could considered a weighted sum of GPs as proposed by Herlands et al. (2016) in which the proportion of each function varies over the domain. We found this suffers from the same restriction. Briefly, if we simplify equations (8) and (9) from Herlands et al. (2016) thus;

$$y(x) = w(x)f(x) + (1 - w(x))g(x),$$

where w is a function describing the weighted summing of the two GP functions f and g. The covariance of this new function is;

$$k(x, x') = w(x)k_f(x, x')w(x') + (1 - w(x))k_g(x, x')(1 - w(x')).$$

If the two points x and x' are within the same regime, then $w(x)$ and $w(x')$ will be roughly equal. However if the two points are from different regions (e.g. if x were in a long-lengthscale region, while x' is associated with a short lengthscale) the products $w(x)w(x')$ or $(1 - w(x))(1 - w(x'))$ will be very small, leading to a small covariance between points across the two regimes, even if those points are nearby.

In summary predictions in the outlier regions will not be supported by training points within the short-lengthscale domain, undermining this as an approach to give more support to outlying regions.

6.4 Classification: Defaulting example

To illustrate the classification method, we use the Home Equity Loans (HEL) data set (Scheule et al., 2017). We start with a one-dimensional example, using as an input the

number of delinquent credit lines (DELINQ). Realistically this is likely to be a feature that an individual would prefer to remain private. We will however leave the provision of input privacy for future work. Instead we assume that it is just the outputs (whether they have defaulted, BAD) that we wish to keep private.

We bias sample two hundred points from the HEL data set to approximately balance both the DELINQ input and the BAD outputs. The upper part of figure 6 indicates the frequencies for each category. We use an EQ kernel (with a lengthscale of 4 credit lines). The middle part of the figure shows, using a violin plot, the (1,0.01)-DP predicted probabilities associated with each possible delinquency count between zero and fourteen. The black crosses indicate the non-DP predicted probabilities. At low counts the risk of default is low but increases with greater numbers of delinquent credit lines. Returning to the distribution of DP posterior means, as the amount of data decreases (over five or six credit lines) we begin to see divergence between the non-DP mean (black crosses) and the means produced with DP noise added during estimation. Specifically, the actual mean remains relatively confident (above 70%), but the DP mean (with noise added) becomes spread across almost the whole range, with concentrations at the extremes due to the squashing effect of the link function.

6.5 Classification: 2d example

To very briefly demonstrate the classification algorithm operating in higher dimensions and to illustrate a slightly more convoluted function, we simulated a simple two-dimensional training set of 200 points consisting of a series of noisy diagonal binary class stripes (Figure 7). Using an EQ kernel with lengthscale=3.5. The figure also shows predictions from 25 separate (1,0.01)-DP-classification computations. A gradient in the concentration of data has been included in the simulated training data. Near the bottom of the image the concentration is high and the DP samples have little variation. Near the training data's class boundaries the predictions are nearer 0.5 (less confident) indicated by smaller dots (e.g. at location [3,2]). Towards the top of the plot (and to an extent around the edge) the training data becomes scarce, leading to greater DP noise manifesting itself as a mix of highly confident predictions for both classes. The scale of the noise could, in future, be included in the test point uncertainty reported.

The method used runs the optimisation step (finding $\hat{f}$) of the Laplace approximation only once. We briefly tried dividing the privacy budget between two iterations, but found the result was considerably worse. We used a test set spaced over the 10×10 grid (consisting of 50 of each class) and computed 25 different DP noise results. We found the accuracy fell from 69% to 51% with the second iteration, suggesting that the additional noise required to ensure privacy in two steps of optimisation far outweighs the potential benefits of an improved estimate of $\hat{f}$. The accuracy without DP noise was 81% (with 20 iterations of the optimiser). It should be noted that better classifiers exist for the non-DP case, and the choice of hyperparameters was deliberately chosen to aid the DP prediction potentially at the cost of the non-DP prediction's accuracy.

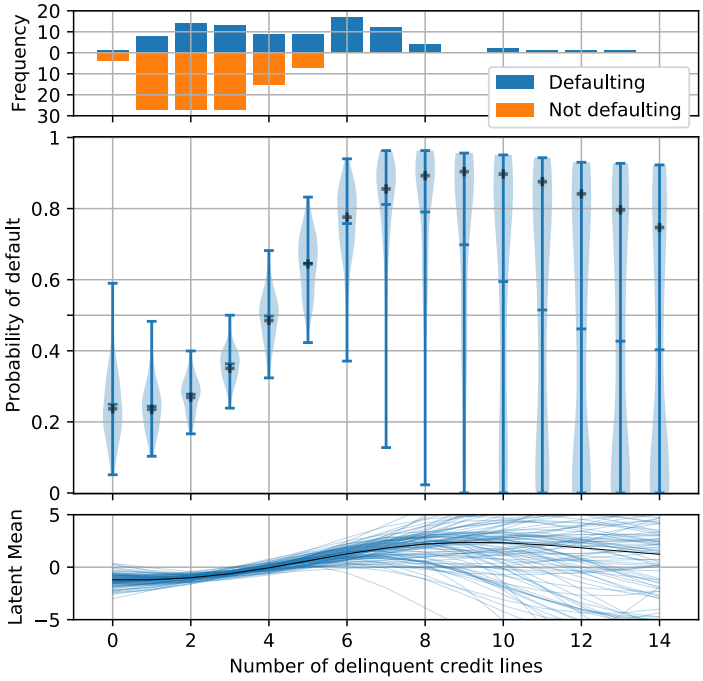

Figure 6: Home Equity Loans data set classification example. A single dimension (number of delinquent credit lines, DELINQ) is used to predict the likelihood of a borrower defaulting on a loan. We assume for the purpose of this demonstration that the training inputs, DELINQ are not private, but the default status is. The upper plot shows the frequency of defaulting for each delinquency count, the middle plot shows the distribution of one hundred DP samples. The black crosses indicate the non-DP mean. The lower plot shows the associated DP latent means (non-DP mean in black).

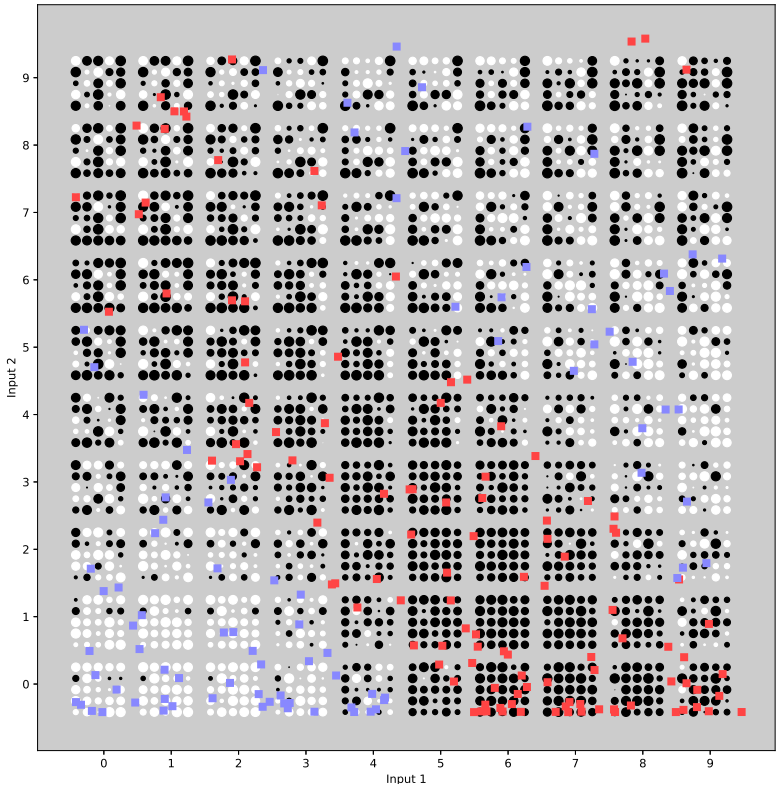

Figure 7: Grouped Hinton plot showing the posterior mean for 25 DP-classification attempts. Each block of circles describes 25 different DP predictions of the transformed posterior mean at each integer location. The white and black circles correspond to values over and under 0.5, respectively. The size of the circle expresses confidence and indicates the distance from 0.5, so the largest circles are either 0 (black) or 1 (white). The same DP sample is at the same relative location in each block. The training inputs are simulated two dimensional values with the blue (+ve) and red (-ve) crosses indicating their location and class. $l_1 = l_2 = 3.5$. 200 iterations were computed for the cloaking algorithm. One iteration of the $\hat{f}$ optimisation was performed. $\varepsilon = 1.0$, $\delta = 0.01$. Sensitivity = 2 (-1 to +1).

6.6 Sparse Classification: MNIST

To demonstrate the DP sparse classification method we turn to a higher dimensional data set. In this problem we try to classify MNIST digits as being in either the low class or high class, {0,1,2,3,4} or {5,6,7,8,9} respectively. The MNIST data points were down-sampled to 15×15 images. We trained the algorithm (using an EQ kernel) with 256 points and tested on 100 other points. Figure 8 demonstrates the impact of both the number of inducing inputs and the lengthscale. The inducing inputs were again chosen using k-means clustering. The lengthscales were all equal and the kernel variance fixed at one. We found that unlike the non-DP results, the (1,0.01)-DP method favoured longer lengthscales and fewer inducing inputs. The former result probably due to the averaging effect longer lengthscales achieve (each individual training point has less influence). The latter result, with fewer inducing inputs providing better predictions is due to the down-weighting of outliers and instead reflects the behaviour of the output around where the data was concentrated. It potentially appears to peak around 21 inducing inputs, and has a marked decline above 40 inducing inputs. We conclude that the inducing inputs help reduce the spurious DP noise one would normally have to introduce to protect the outliers as we found in the regression example in Section 6.2. This data set sits on a relatively low-dimensional manifold due to the correlations between neighbouring pixels and the relative similarity of training data points, hence it was concentrated in such a way that only a few inducing inputs were able to approximate it quite well, not necessarily a feature of all high-dimensional data sets.

In Section 6.5 we briefly investigated whether using just one iteration to compute $\hat{\boldsymbol{f}}$ was a valid trade-off between the additional DP-noise required each iteration vs the potential to improve the estimate of $\hat{\boldsymbol{f}}$. Figure 9 shows why one iteration is likely to be sufficient. We use (1,0.01)-DP noise with an EQ kernel, with $l = 380$. The left figure shows the non-DP optimisation of values in the $\hat{\boldsymbol{f}}$ vector. Notice how quickly the values converge, with 97.8% of the total optimisation taking place in the first iteration. The DP example appears to fail to converge, however the prediction is significantly better than chance (68% correct in this case) as, on average, the $\hat{\boldsymbol{f}}$ vector still has useful signal within it (although contains considerable DP noise).

6.7 !Kung Hyperparameter Selection

For our 1d !Kung example our model has three parameters, the lengthscale, the kernel variance and the likelihood's Gaussian noise variance. We need to select these in such a way that the training data remains private, while the predictions have the lowest SSE possible. Note that for simplicity we use in this section the standard cloaking method without sparse approximation. For this experiment we split the data into two halves to avoid circular analysis; using one half for selecting hyperparameters, and the other for estimating the RMSE achieved if we had used those hyperparameters. The training half is again split in a 5-fold cross-validation step to estimate the SSE for each configuration of hyperparameters. This leaves only 115 points for training. We target, as before, (1,0.01)-DP. The sensitivity for the SSE is computed based on this cross-validation. We can then report the expected SSE of the training set using the probabilities provided by the exponential mechanism. It would be worth matching the size of the training set to the size of the final data set used in a genuine

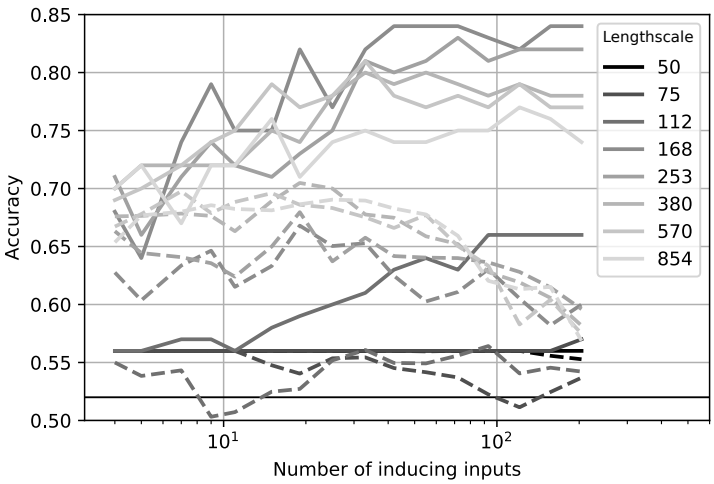

Figure 8: MNIST DP-classification. Graph showing accuracy of sparse GP classifier with DP (dashed lines) or without DP (solid lines). Chance accuracy at 52% indicated by horizontal black line. x-axis indicates number of inducing inputs (between 4 and 204, N=256). Shade indicates lengthscale. Pixel values in the data varied from 0 to 255, so the lengthscale spans this, with black being short lengthscales and light grey, long. For the non-DP classification there is an optimal lengthscale of about 168, and as one might expect, the accuracy improves with more inducing inputs. The DP method on the other hand is optimal with longer lengthscales, presumably because this allows more training points to be averaged. It also peaks at lower numbers of inducing inputs. As discussed for the sparse regression results, this is due to reducing outlier influences. DP points computed from 25 DP model optimisation runs leading to a maximum SE of 1.86% accuracy. Associated CIs omitted for clarity. The number of inducing inputs was tested at sixteen exponentially spaced values (4 to 200).

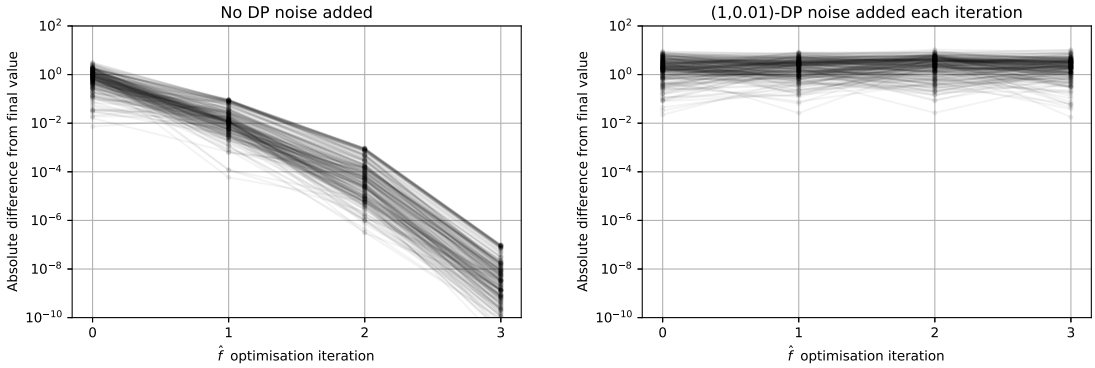

Figure 9: Graphs showing iterations in the optimisation of $\hat{\boldsymbol{f}}$, with each line reflecting the absolute difference from the optimum for each element of the vector. Left plot, no DP noise; right plot, $(1, 0.01)$-DP noise added each iteration. The model being optimised was for the MNIST example described in Section 6.6, with $l = 380$ and 16 inducing points.

application (or alternatively modify the scale of the DP noise to simulate the perturbation had the same number of training points been in both) to ensure that the utilities of the different models reflect those experienced when applied.

In this experiment we search exponentially the three parameters (lengthscale 1, 5, 25, 125, 625 years; Gaussian noise variance 0.2, 1, 5, 25 cm^2; Kernel variance 1, 5, 25, 125 cm^2). Figure 10 shows the RMSEs, with each bar's width representing the probability of selecting that configuration. The full table of RMSEs and probabilities is in the supplementary. The configuration that achieved the lowest error in the training set was found to be with lengthscale, 25 years; Gaussian noise variance, 25 cm^2; kernel variance, 1.0 cm^2; with a RMSE of 14.55cm. The long lengthscale is particularly relevant here as, unlike in Section 6.1, we only have 115 training points, so averaging over the data provides the best compromise between an accurate fit and avoiding excessive DP noise. If we were to simply select this configuration we would be leaking private data through the computation of the SSEs. To avoid this we select a configuration at random with probability calculated using the exponential mechanism ($\varepsilon = 1$). Although the above configuration had the least training RMSE it will still only get selected with a probability of 0.0244, but this is almost twice the $1/80$ average (given we tested 80 configurations). The expected RMSE (from the weighted sum of the RMSEs of the available configurations) is 19.02cm. While the mean of the 80 configurations is 87.05cm. Clearly picking a configuration using the exponential mechanism is better than randomly choosing, but not as good as the best configuration. We used up $\varepsilon = 2$ DP, overall ($\varepsilon = 1$ for the exponential mechanism and $\varepsilon = 1$ for the DP GP regression). The trade-off in how this budget is spent is left for future experimentation.

An interesting result is in the effect of the level of privacy on the selection of the lengthscale. This is demonstrated in the distribution of probabilities over the lengthscales when we adjust ε. Figure 11 demonstrates this effect using the 1d !Kung example. Each column is for a different level of privacy (from none to high) and each tile shows the probability of selecting that lengthscale. For low privacy, short lengthscales are acceptable, but as the pri-

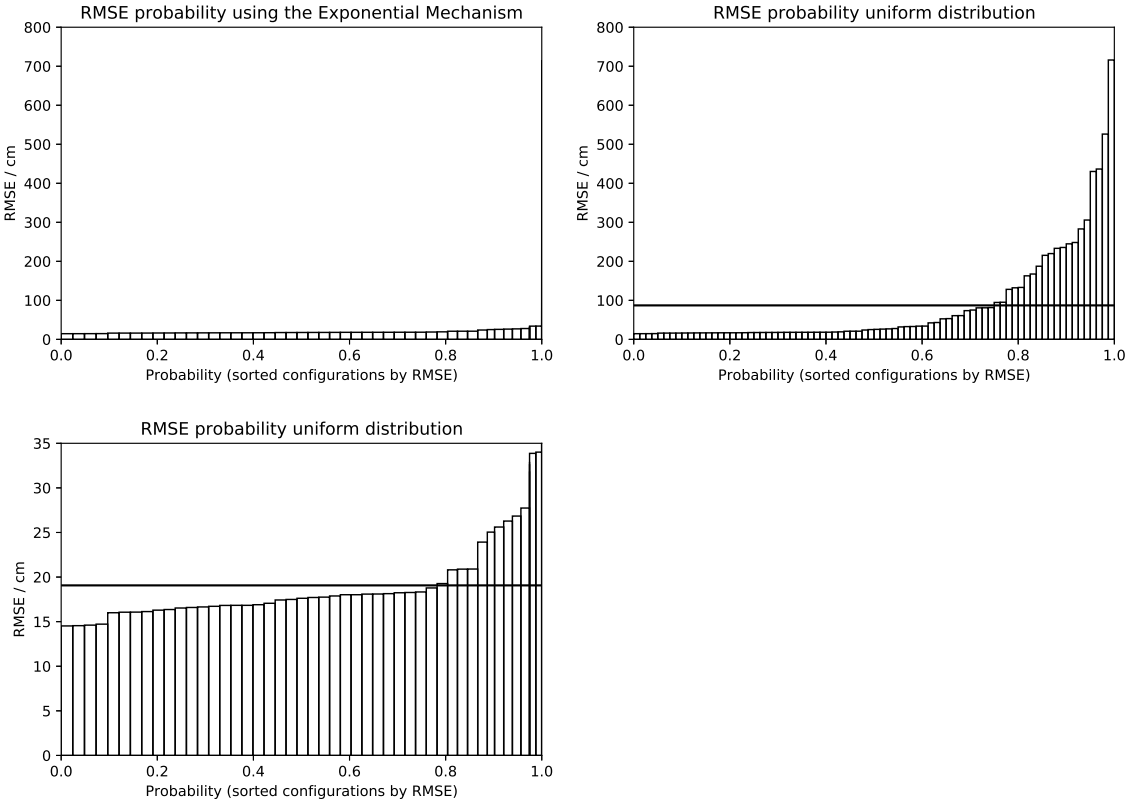

Figure 10: The upper two graphs show hyperparameter configurations sorted by their RMSE on the same scale, to allow easy comparison. On the left is the result using the exponential mechanism and on the right the configurations are given equal probability. The lower figure is simply the upper figure scaled to allow the distribution to be observed. The horizontal lines mark the expected RMSE for the distribution plotted.

vacy increases, averaging over more data by increasing the lengthscale allows us to mitigate the increasing DP noise.

7. Discussion

We have investigated several ways of expanding the range and power of DP Gaussian process models. We first looked at methods for reducing the excessive noise added in outlier regions of the data set, an issue we showed would particularly need addressing if one wanted to do inference at increased dimensionality. Specifically we considered both sparse approximations and methods to manipulate the lengthscale. We found that the latter failed to achieve the expected gains as the data within the short-lengthscale high-density regions was largely uncorrelated with the outliers, leaving them with little support. However the sparse method, using inducing inputs, was able to substantially reduce the DP noise without compromising the DP guarantees.

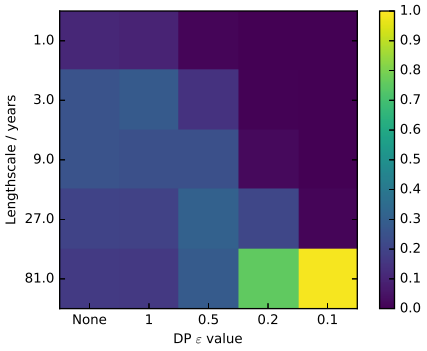

Figure 11: Using the 1d !Kung data set. Effect of varying the differential privacy parameter, ε, on the likelihood of selecting each lengthscale. Colour indicates probability of parameter selection. With low privacy, a short lengthscale is appropriate which allows the GP to describe details in the data. With high privacy, a longer lengthscale is required, which will average over large numbers of individual data points. Used the whole data set with 5-fold cross-validation.

To describe these approaches simply, they effectively down-weight or 'ignore' the outliers to reduce their contribution to the model predictions, which intuitively leads to less sensitivity to these training points. It should be borne in mind however that we are exchanging one penalty suffered by outliers, that their privacy is easily compromised, for another, that they will be largely ignored when the model makes predictions. There is no 'free-lunch'. That said these approaches empirically provide improvements on prediction accuracy.

Both sparse and nonstationary methods require additional parameters to be selected. Specifically, the sparse method requires the number and placement of inducing inputs to be chosen, while the variable lengthscale method requires a lengthscale function to be defined. In this case the inducing inputs were placed using simple k-means clustering. The lengthscale function required more attention and care. Note that in practice selecting a function with awareness of the data will lead to unintentional privacy-loss. Thus, from a practical, privacy perspective the sparse, inducing-input method appears to offer a safer easier to apply solution, which also performs substantially better. We applied a similar sparse-approximation to the problem of classification, for a larger scale data set and found it also improved the results if we reduced the number of inducing inputs. It also confirmed the DP method could function with high-dimensional data, although the structure of the points within the domain may be important. We hypothesise that the method is most effective when the data lies on a relatively low-dimensional manifold, or concentrated into relatively few locations in the input domain. An interesting area for future work would be to investigate which approximations achieve better DP noise results. These might not be the approximations that achieve the most accurate non-DP posterior distributions. In general, it could be very effective if one were to build a formalism around these approaches to non-stationary regression, which would improve understanding and allow the choice of non-linearity to be optimised for the DP problem.

A brief note on the placement of the inducing inputs. We could use a more traditional method that minimises a KL divergence to find the optimum locations. However this re-

quires 'peeking' at the outputs. Thus we would need to use a gradient descent method that is privacy-aware (Abadi et al., 2016; Song et al., 2013) and bound the sensitivity of the training data on these gradients. More fundamentally, placement using such a standard sparse approximation method would place the inducing inputs in potentially very suboptimal locations in the DP situation. For example, a single outlier often is given its own inducing input when trying to minimise the KL divergence (if there's sufficient inducing inputs available) which will lead to massive DP noise added in the vicinity of this outlier. Recently Manousakas et al. (2020) proposed the use of a 'pseudocoreset' which has similarities to the inducing points in this paper. They use DP stochastic optimisation so it might be vulnerable to the outlier problem if their method was applied to GPR, however they use a variational approximation which one might be able to define such that it incorporates the added DP noise, potentially avoiding the outlier issue.

We next investigated DP GP classification. We made the predictions DP in the output by altering the internal iterative update that finds the mode of the posterior. It was found however that updating this more than once reduces the classification accuracy, as the additional DP noise required to span two updates outweighs the possible benefit from a better estimate of the mode. This might change with more training data, or additional dimensions. The latent function's DP mean behaves as in the regression case and contains more variance in outlier regions, even though the mechanism for the application of DP is somewhat different. The unexpected utility of the cloaking mechanism to this problem suggests that the cloaking method may have other applications, in which a vector of training data is linearly transformed. In future work the Gaussian noise of the DP should be included in the computation of the approximate posterior (through the squashing, logistic function) rather than just use the variance of the latent function, as this will allow this component of the uncertainty to be included in any prediction. We excluded it from the posterior for clarity in the plots and results (allowing the DP and GP uncertainty to be distinguished).

We finally looked at the issue of hyperparameter optimisation. We found that a crude grid search combined with the DP exponential mechanism provides a simple but effective method for selecting hyperparameters. Its efficacy may be down to the relatively low sensitivity of the RMSE to parameter selection, with many configurations resulting in similar error rates. Figure 10 demonstrates this, with nearly half the configurations having a RMSE lying within 30% of the best. The number of grid entries will clearly grow exponentially as the number of hyperparameters increases. Future work should investigate DP-Bayesian optimisation or maximum-likelihood DP-gradient ascent for hyperparameter optimisation. We also could use this approach to select the number of inducing points and potentially some aspects of their locations.

One potential future direction is to look at how one might make repeated queries using the cloaking framework. If the values of the DP-noise are stored on the server, one could condition on these in future queries, to generate predictions while using additional privacy budget efficiently. For example if the future query test point lies very close to a previously released test point, the DP noise variance for the new test point will be (relatively) very small. Thus, a very small value of ϵ could be used, and the algorithm would still produce a prediction with a reasonable level of DP noise. It would be interesting in future to explore both this approach and the potential for adding new training data and still produce predictions in an efficient way.

In summary, we have presented three tools to extend the use of differential privacy for Gaussian process models. First, the sparse approximation works by effectively reducing the effect of the outliers, allowing the model to make predictions in both dense and outlier regions. This becomes particularly important at higher dimensions. Second we devised a method for making a GP classification model differentially private, by altering its internal optimisation step. We found it was able to make predictions with a useful accuracy and with a relatively small number of training points. Finally we suggest a method for hyperparameter optimisation based on a straightforward grid search combined with the differentially private exponential mechanism.

Acknowledgments

This work has been supported by the Engineering and Physical Research Council (EPSRC) Research Project EP/N014162/1. We also thank Wil Ward and Fariba Yousefi for their assistance and suggestions.

Appendix A. Differentially Private Hyperparameter Selection

Table 2 reports the probability and RMSE for the combinations used in the hyperparameter optimisation exercise in Section 6.7.

LS	σ^2	Kernel var	Prob	RMSE		LS	σ^2	Kernel var	Prob	RMSE	
1.0	0.2	1.0	0.0000	235.52		25.0	5.0	1.0	0.0231	16.36	
1.0	0.2	5.0	0.0000	436.54		25.0	5.0	5.0	0.0210	20.89	
1.0	0.2	25.0	0.0000	526.11		25.0	5.0	25.0	0.0177	26.84	
1.0	0.2	125.0	0.0000	715.99		25.0	5.0	125.0	0.0000	31.85	
1.0	1.0	1.0	0.0000	132.11		25.0	25.0	1.0	0.0244	14.55	***
1.0	1.0	5.0	0.0000	244.78		25.0	25.0	5.0	0.0233	16.13	
1.0	1.0	25.0	0.0000	305.81		25.0	25.0	25.0	0.0205	20.81	
1.0	1.0	125.0	0.0000	430.29		25.0	25.0	125.0	0.0180	27.74	
1.0	5.0	1.0	0.0000	73.59		125.0	0.2	1.0	0.0233	16.53	
1.0	5.0	5.0	0.0000	162.80		125.0	0.2	5.0	0.0223	18.02	
1.0	5.0	25.0	0.0000	219.82		125.0	0.2	25.0	0.0219	19.27	
1.0	5.0	125.0	0.0000	283.01		125.0	0.2	125.0	0.0211	20.90	
1.0	25.0	1.0	0.0190	25.61		125.0	1.0	1.0	0.0236	16.07	
1.0	25.0	5.0	0.0000	60.73		125.0	1.0	5.0	0.0233	16.29	
1.0	25.0	25.0	0.0000	215.35		125.0	1.0	25.0	0.0222	18.24	
1.0	25.0	125.0	0.0000	248.26		125.0	1.0	125.0	0.0225	18.78	
5.0	0.2	1.0	0.0000	75.13		125.0	5.0	1.0	0.0226	17.43	
5.0	0.2	5.0	0.0000	94.60		125.0	5.0	5.0	0.0236	16.00	
5.0	0.2	25.0	0.0000	128.16		125.0	5.0	25.0	0.0230	16.72	
5.0	0.2	125.0	0.0000	233.28		125.0	5.0	125.0	0.0228	17.62	
5.0	1.0	1.0	0.0000	52.59		125.0	25.0	1.0	0.0225	18.10	
5.0	1.0	5.0	0.0000	80.72		125.0	25.0	5.0	0.0227	17.49	
5.0	1.0	25.0	0.0000	95.11		125.0	25.0	25.0	0.0235	16.06	
5.0	1.0	125.0	0.0000	167.44		125.0	25.0	125.0	0.0232	16.65	
5.0	5.0	1.0	0.0131	33.87		625.0	0.2	1.0	0.0230	16.83	
5.0	5.0	5.0	0.0000	133.09		625.0	0.2	5.0	0.0242	14.72	**
5.0	5.0	25.0	0.0000	81.38		625.0	0.2	25.0	0.0232	17.06	
5.0	5.0	125.0	0.0000	187.44		625.0	0.2	125.0	0.0225	17.75	
5.0	25.0	1.0	0.0222	18.27		625.0	1.0	1.0	0.0224	18.09	
5.0	25.0	5.0	0.0122	34.01		625.0	1.0	5.0	0.0231	16.90	
5.0	25.0	25.0	0.0000	53.16		625.0	1.0	25.0	0.0242	14.61	**
5.0	25.0	125.0	0.0000	80.85		625.0	1.0	125.0	0.0231	16.59	
25.0	0.2	1.0	0.0151	25.04		625.0	5.0	1.0	0.0224	18.14	
25.0	0.2	5.0	0.0000	32.92		625.0	5.0	5.0	0.0225	18.02	
25.0	0.2	25.0	0.0000	60.83		625.0	5.0	25.0	0.0230	16.83	
25.0	0.2	125.0	0.0000	43.17		625.0	5.0	125.0	0.0241	14.52	*
25.0	1.0	1.0	0.0200	23.92		625.0	25.0	1.0	0.0226	17.71	
25.0	1.0	5.0	0.0181	26.28		625.0	25.0	5.0	0.0223	18.33	
25.0	1.0	25.0	0.0000	32.66		625.0	25.0	25.0	0.0224	17.88	
25.0	1.0	125.0	0.0000	42.26		625.0	25.0	125.0	0.0231	16.82	

Table 2: Hyperparameter search using the exponential mechanism. The top four parameter configurations that are most likely to be chosen are marked with asterisks. The probability is based on the SSE from the training fold while the RMSE is from the test fold. Many probabilities are exactly zero due to the thresholding of the SSEs.

Appendix B. Example using a more complex kernel

In Figure 12 we use the 1d !Kung data, extrapolating rather unrealistically to offer a demonstration with a more complex kernel. The plots demonstrate that the cloaking methods (standard and sparse) work with other kernels. Indeed the methods are relatively agnostic, and simply process the resulting cloaking matrix. The benefit of the inducing point approximation may vary when the kernel itself causes outliers to become more correlated with other data.

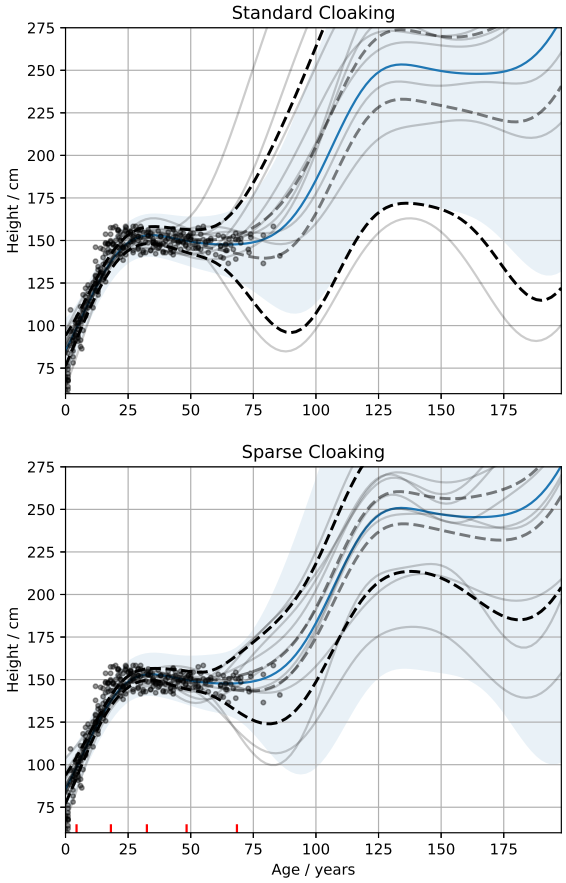

Figure 12: The cloaking method and its sparse version work with arbitrary kernels. In this plot we use the 1d !Kung data set, but this time use a Bias+Linear+Periodic kernel (period=100 years). Solid blue lines, posterior non-DP means of the GPs; grey lines, DP samples; Black and grey dashed lines, standard error and $\frac{1}{4}$ standard error confidence intervals for DP noise respectively; blue area, non-DP GP posterior variance (excluding noise). $\varepsilon = 1$, $\delta = 0.01$, $\Delta = 100$ cm. Upper plot, standard cloaking method. Lower plot, cloaking method, using five inducing inputs (placed using k-means clustering).

References

Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. ACM, 2016.

Armita Afsharinejad and Neil Hurley. Performance analysis of a privacy constrained kNN recommendation using data sketches. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 10–18. ACM, 2018.

Kamalika Chaudhuri and Staal A Vinterbo. A stability-based validation procedure for differentially private machine learning. In *Advances in Neural Information Processing Systems*, pages 2652–2660, 2013.

Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(Mar):1069–1109, 2011.

Richard Comont and Stephanie Miles. Beewalk annual report 2019. *Bumblebee Conservation Trust*, 2019.

Christos Dimitrakakis, Blaine Nelson, Zuhe Zhang, Aikaterini Mitrokotsa, and Benjamin IP Rubinstein. Differential privacy for Bayesian inference through posterior sampling. *Journal of Machine Learning Research*, 18(Jan):343–381, 2017.

Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

Sam Fletcher and Md Zahidul Islam. Differentially private random decision forests using smooth sensitivity. *Expert Systems with Applications*, 78:16–31, 2017.

Srivatsava Ranjit Ganta, Shiva Prasad Kasiviswanathan, and Adam Smith. Composition attacks and auxiliary information in data privacy. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 265–273. ACM, 2008.

Mark N Gibbs. *Bayesian Gaussian processes for regression and classification*. PhD thesis, University of Cambridge, 1998.

Rob Hall, Alessandro Rinaldo, and Larry Wasserman. Differential privacy for functions and functional data. *Journal of Machine Learning Research*, 14(Feb):703–727, 2013.

William Herlands, Andrew Wilson, Hannes Nickisch, Seth Flaxman, Daniel Neill, Wilbert Van Panhuis, and Eric Xing. Scalable Gaussian processes for characterizing multidimensional change surfaces. In *Artificial Intelligence and Statistics*, pages 1013–1021, 2016.

Howell, N. Data from a partial census of the !Kung San, dobe. 1967-1969, 1967.

Geetha Jagannathan, Krishnan Pillaipakkamnatt, and Rebecca N Wright. A practical differentially private random decision tree classifier. In *IEEE International Conference on Data Mining Workshops*, pages 114–121. IEEE, 2009.

Matt Kusner, Jacob Gardner, Roman Garnett, and Kilian Weinberger. Differentially private Bayesian optimization. In *International Conference on Machine Learning*, pages 918–927, 2015.

Jing Lei. Differentially private m-estimators. In *Advances in Neural Information Processing Systems*, pages 361–369, 2011.

Haoran Li, Li Xiong, Lucila Ohno-Machado, and Xiaoqian Jiang. Privacy preserving RBF kernel support vector machine. *BioMed research international*, 2014.

Qinbin Li, Zhaomin Wu, Zeyi Wen, and Bingsheng He. Privacy-preserving gradient boosting decision trees. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 784–791, 2020.

Jingcheng Liu and Kunal Talwar. Private selection from private candidates. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 298–309, 2019.

Dionysios Manousakas, Zuheng Xu, Cecilia Mascolo, and Trevor Campbell. Bayesian pseudocoresets. *preprint, Cambridge*, 2020.

Christopher J Paciorek and Mark J Schervish. Nonstationary covariance functions for Gaussian process regression. In *Advances in Neural Information Processing Systems*, pages 273–280, 2004.

Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6(Dec): 1939–1959, 2005.

Benjamin Rubinstein, Peter Bartlett, Ling Huang, and Nina Taft. Learning in a large function space: Privacy-preserving mechanisms for SVM learning. *Journal of Privacy and Confidentiality*, 4(1):65–100, 2012.

Harald Scheule, Daniel Rösch, and Bart Baesens. *Credit Risk Analytics: The R Companion*. Create Space Independent Publishing Platform, 2017.

Michael Thomas Smith, Mauricio A Álvarez, Max Zwiessele, and Neil D Lawrence. Differentially private Gaussian processes. *Artificial Intelligence and Statistics*, 1050:30, 2018.

Alex J Smola and Peter L Bartlett. Sparse greedy Gaussian process regression. In *Advances in Neural Information Processing Systems*, pages 619–625, 2001.

Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, pages 1257–1264, 2006.

Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*, pages 245–248. IEEE, 2013.

Latanya Sweeney. Weaving technology and policy together to maintain confidentiality. *The Journal of Law, Medicine & Ethics*, 25(2-3):98–110, 1997.

Latanya Sweeney. Only you, your doctor, and many others may know. *Technology Science*, 2015092903(9):29, 2015.

Anthony Tockar. Riding with the stars: Passenger privacy in the NYC taxicab dataset. `https://agkn.wordpress.com/2014/09/15/riding-with-the-stars-passenger-privacy-in-the-nyc-taxicab-dataset`, 2014.

Jaideep Vaidya, Basit Shafiq, Anirban Basu, and Yuan Hong. Differentially private naive Bayes classification. In *IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volume 1, pages 571–576. IEEE, 2013.

Christopher KI Williams and Carl Edward Rasmussen. Gaussian processes for machine learning. *MIT Press*, 2(3):4, 2006.

Kan Yang, Kuan Zhang, Ju Ren, and Xuemin Shen. Security and privacy in mobile crowdsourcing networks: challenges and opportunities. *IEEE communications magazine*, 53(8): 75–81, 2015.

Jun Zhang, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, and Marianne Winslett. Functional mechanism: regression analysis under differential privacy. *Proceedings of the VLDB Endowment*, 5(11):1364–1375, 2012.

Tianqing Zhu, Yongli Ren, Wanlei Zhou, Jia Rong, and Ping Xiong. An effective privacy preserving algorithm for neighborhood-based collaborative filtering. *Future Generation Computer Systems*, 36:142–155, 2014.